Winter 12-29-2006

The Unbalanced Classification Problem: Detecting Breaches in Security

Paul Evangelista
United States Military Academy, paul.evangelista@westpoint.edu

THE UNBALANCED CLASSIFICATION PROBLEM: DETECTING BREACHES IN SECURITY

By

Paul F. Evangelista

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Decision Sciences and Engineering Systems

Approved by the
Examining Committee:

Mark J. Embrechts, Thesis Adviser

Boleslaw K. Szymanski, Thesis Adviser

Joseph G. Ecker, Member

William A. Wallace, Member

Robert H. Kewley, Jr., Member

Rensselaer Polytechnic Institute
Troy, New York

November 2006
(For Graduation December 2006)

THE UNBALANCED CLASSIFICATION PROBLEM: DETECTING BREACHES IN SECURITY

By

Paul F. Evangelista

An Abstract of a Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Major Subject: Decision Sciences and Engineering Systems

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Examining Committee:

Mark J. Embrechts, Thesis Adviser
Boleslaw K. Szymanski, Thesis Adviser
Joseph G. Ecker, Member
William A. Wallace, Member
Robert H. Kewley, Jr., Member

Rensselaer Polytechnic Institute
Troy, New York

November 2006
(For Graduation December 2006)

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

There are a tremendous number of people and organizations who supported this work. From a technical standpoint, I thank Mike Kupferschmidt for motivating me to use LaTeXand Harriet Borton for sharing her brilliant LaTeXexpertise. Professor Al Wallace started me on this work a long time ago by sharing his trademark common sense and no nonsense advice on how to proceed. The United States Army entrusted me and provided me with the opportunity to complete this work. The Department of Systems Engineering at the United States Military Academy, West Point, played an instrumental role with guidance and unwavering support of this work. More specifically, the leadership of the Systems Engineering Department and climate that they foster enabled this work. Robert Kewley provided crucial guidance early during the course of this work. Most importantly, he was a tremendous trail blazer who led the way and generously showed me the way down the trail to finish this work. My advisers, Professors Mark Embrechts and Bolek Szymanski, shared substantial amounts of time on a consistent basis demonstrating patience and trust in my work. As my primary adviser, Professor Embrechts provided an unquestionable touch of brilliance to this work but more importantly impressed me as a sincere, selfless leader of his students.

The support and love from my family during this work will never be forgotten. I thank my parents for passing along precious values that I consider as bedrock for all that I do. My beautiful children Alex and Jacqueline inspire me every day. They are the true meaning of happiness. My wife, Heather, supported this work from start to finish with unrivaled generosity, love, and loyalty. She provided me with counsel, tolerated the seemingly endless hours of "tinkering", gladly had food on the table for me every night, and even gave me the necessary kick I occasionally needed to get moving. Bliss and success occurred in my life when I married her.

Lastly to the source of my strength and my being, my Lord and my Faith continue to provide tremendous inspiration, guidance, and clarity - especially when nothing seems clear. Without this, I would unquestionably be lost.

ABSTRACT

This research proposes several methods designed to improve solutions for security classification problems. The security classification problem involves unbalanced, high-dimensional, binary classification problems that are prevalent today. The imbalance within this data involves a significant majority of the negative class and a minority positive class. Any system that needs protection from malicious activity, intruders, theft, or other types of breaches in security must address this problem. These breaches in security are considered instances of the positive class. Given numerical data that represent observations or instances which require classification, state of the art machine learning algorithms can be applied. However, the unbalanced and high-dimensional structure of the data must be considered prior to applying these learning methods. High-dimensional data poses a "curse of dimensionality" which can be overcome through the analysis of subspaces. Exploration of intelligent subspace modeling and the fusion of subspace models is proposed. Detailed analysis of the one-class support vector machine, as well as its weaknesses and proposals to overcome these shortcomings are included. A fundamental method for evaluation of the binary classification model is the receiver operating characteristic (ROC) curve and the area under the curve (AUC). This work details the underlying statistics involved with ROC curves, contributing a comprehensive review of ROC curve construction and analysis techniques to include a novel graphic for illustrating the connection between ROC curves and classifier decision values. The major innovations of this work include synergistic classifier fusion through the analysis of ROC curves and rankings, insight into the statistical behavior of the gaussian kernel, and novel methods for applying machine learning techniques to defend against computer intrusion detection. The primary empirical vehicle for this research is computer intrusion detection data, and both host-based intrusion detection systems (HIDS) and network-based intrusion detection systems (NIDS) are addressed. Empirical studies also include military tactical scenarios.

CHAPTER 1
INTRODUCTION AND INNOVATIONS

During the air defense battles of World War II, the value of radar and signal detection grew at an explosive rate. Radar techniques were relatively primitive and required significant human interaction and interpretation. The essence of the problem was simple. Radar detected incoming aircraft. A radar operator needed to be able to distinguish between friendly and enemy aircraft. Identifying a friendly aircraft as enemy (a false positive), created an expensive sequence of drills and defensive responses. This could potentially subject the inbound friendly aircraft to friendly fire. There was also the danger of not alerting when actual enemy aircraft were inbound, a false negative. Radar represents one of the earliest signal detection problems which required humans to interact and interpret a technical measure, the radar signal, with the overall goal of classifying the observation as one of two classes. This is a binary classification problem. In order to measure the effectiveness of radar operators, the military recorded the performance of these radar operators. This performance measure became known as the radar receiver operating characteristic illustrated on the receiver operating characteristic (ROC) curve [53]. Radar became one of the earliest applications of signal detection theory. After World War II, atomic weapons boosted the importance of air defense. In the 1950s tremendous research efforts, such as the MIT led Project Charles, communicated the vast problems and gaps that existed in national air defense. The final report of Project Charles, originally a classified document, emphasized that in order to improve air defense the program would need significant manpower and a deliberate layered detection strategy in order to overcome costly false positives [103].

Much of the enemy aircraft threat existed in remote areas where vast expanses, such as oceans or harsh northern territories further constrained the nation's ability to man an adequate air defense system. This marked the beginning of a long quest to automate signal detection. Naka and Ward explain the history of air defense coverage across Alaska and Canada and the critical need for an automated system

that could defend this enormous territory [115].

Although ROC curves and automated signal detection has matured immensely since its inception over half a century ago, there is a continued effort to improve automation and vast use of ROC curves to support various types of binary decisions. This dissertation contributes several novel methods that involve the automation and accuracy of binary prediction models. Although the applications presented in this thesis do not involve radar and air defense, many similar threads exist between early research of automated binary decision making and the novelties included in this dissertation. Today's binary classification problems have increased complexity and dimensionality, however today's binary prediction models and computing resources provide tools and leverage necessary to tackle the problems. There is still a continued quest to automate these systems, reduce false positives, and simply improve overall accuracy. In many ways, this quest is no different from the quest that Naka and Ward discuss when they explain one of the original quests for automated signal detection that occurred over half a century ago [115].

1.1 Statement of the Problem

A novel framework for solving the high-dimensional unbalanced binary classification problem in both the supervised and unsupervised domain is proposed. This type of classification problem will be referred to as the **security classification problem**.

A typical security domain abounds with healthy instances or observations. The healthy instances in a security problem will always comprise the majority, and hence are called the majority class. The unhealthy, or positive instances are the minority class.

1.2 Notation and Formulation of the Security Classification Problem

The formulation of the security problem assumes there is a given data set $\mathbf{X} \in \mathbb{R}^{N \times m}$. $\mathbf{X}$ contains N instances or observations, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, where $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$. There are m variables to represent every instance. For every instance there is a label

or class, $y_i \in \{-1, +1\}$. The unbalanced nature of this problem indicates a prevailing negative or healthy class and minimal instances of the positive or unhealthy class.

A fundamental problem within the security classification problem is that increased dimensionality, especially if it contains noisy data, can degrade learning. For classification problems that assume the data is balanced and plentiful, there are feature extraction techniques designed to effectively reduce dimensionality without degrading performance [67, 76], however this is not the case with the security classification problem. Managing high-dimensional data is not as simple if the classes are unbalanced. It is even more difficult when there are no instances of the positive class or unknown instances of the positive class in the training data.

The underlying premise of this research is that given a security classification problem, improved performance can be achieved through a divide and conquer approach that involves analyzing the subspaces within $\mathbf{X}$.

1.3 Objectives of the Research

This research includes three objectives:

1. *Create a general framework for improving the accuracy of solutions for security classification problems.* This framework will include specific models that are particularly applicable for security classification problems, however more importantly the framework will provide procedural knowledge for solving security classification problems. This framework will also expose a taxonomy for security classification problems to enhance a practitioner's ability to quickly identify the type and nature of problem he is facing.

2. *Examine computer intrusion detection data as the primary empirical vehicle for this research.* Intrusion detection is an excellent example of a security classification problem. Intrusion detection datasets which stress state-of-the-art learning models are available, and additionally this is an active and constantly changing field of research.

3. *Utilize and enhance state-of-the-art learning methods for solving security classification problems.* Unsupervised learning is a major component of this re-

search, with a particular emphasis on one-class support vector machines (SVM) which are suited exceptionally well for solving security classification problems. Learning models will focus on kernel methods. These advanced learning methods will be combined with multivariate statistical analysis and dimension reduction techniques to address problems within the learning models that have not been solved and tailor these models for security classification problems.

1.4 Hypotheses

Several hypotheses that seek to characterize the security classification problem have been created. The purpose of itemizing the hypotheses is to clearly indicate unsolved problems and the chapters within this dissertation that address these unsolved problems.

1. *Kernel-based pattern recognition suffers from a curse of dimensionality.* The underlying theory and fundamentals that prove this hypothesis as true are shown in this chapter. This is also shown experimentally in section 2.2.3.3.

2. *Subspace modeling and fusion of models with fuzzy logic aggregators can improve accuracy.* This is perhaps the most important finding of this dissertation. Chapter 6 provides theoretical and empirical proof that class imbalance is a critical additional parameter that must be included when fusing models. T-norms (minimum, algebraic product, etc.) have consistently demonstrated superior performance as a subspace aggregator when there is a minority positive class. Modeling in randomly selected subspaces consistently outperforms models constructed from the entire set of dimensions. Chapters 5 and 6 detail the efforts that prove this hypothesis as true.

3. *The general problem of the curse of dimensionality can be systematically solved with intelligent subspace modeling and model fusion.* This is an explored but unproven hypothesis. Selecting the right subspaces is a difficult problem which has been approached with several alternatives. Chapter 5 includes several exploratory efforts that aim to create intelligent subspace models.

4. *Statistical analysis of the gaussian kernel provides an opportunity for automated kernel tuning for the one class SVM.* The nonlinear patterns recognized by the gaussian kernel make this kernel a favorite amongst practitioners. However, this kernel contains an additional free parameter that requires additional tuning and validation. Chapter 7 provides novel insight into the statistical behavior of the gaussian kernel. Optimization of certain statistical measures of the gaussian kernel present opportunities for automated tuning. The chapter proposes a heuristic that illustrates the use of these statistics to automate the tuning of the gaussian kernel. Promising empirical results support use of the heuristic as an automated tuning method.

1.5 Innovative Progress to Date

This section highlights some of the novel contributions of this research. Each of the following subsections summarizes an innovative approach that has been applied during the course of this research.

1.5.1 Pursuit of the Synergistic ROC Curve

Given that the security problem involves binary classification, Receiver operating characteristic (ROC) curves will provide the primary measure of effectiveness of the models. The ROC curve plots the false positive rate on the x-axis and the true positive rate on the y-axis. Given that both of these rates have minimal values of zero and maximal values of unity, the area under the ROC curve also ranges between zero and one. This area is an excellent scalar measure of binary classification, and it also involves a probabilistic interpretation. This area is the probability that the classifier will correctly rank a negative instance lower than a positive instance. The pursuit throughout this research is to develop techniques that improve the area under the ROC curve for security classification problems, both in the supervised and unsupervised domain. Consistent with the divide and conquer approach, subspaces of $\mathbf{X}$ play a critical role throughout this research. Every subspace could provide an independent model, and each independent model would create an ROC curve. An ongoing pursuit involves combining these independent models such that synergistic

improvement of the ROC curve occurs.

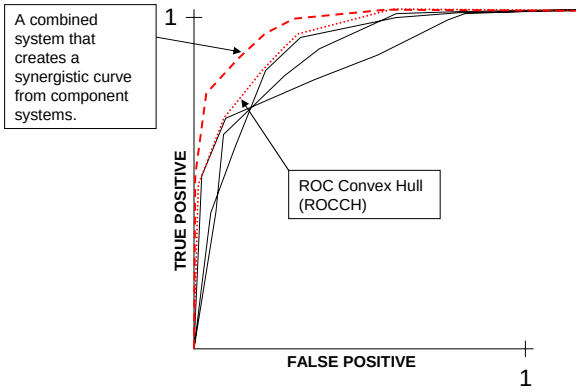

Figure 1.1: Illustration of synergistic ROC curve vs. ROCCH

1.5.2 The Relationship between ROC Curves and Decision Values

Chapter 3 provides a summary on how to construct ROC curves from ranks. The chapter includes several algorithms used for construction, with an additional discussion regarding the analysis of the partial area under the ROC curve. Most of the chapter is not novel information, however the chapter does provide a useful survey regarding ROC curve construction. The one novel aspect of the chapter involves a graphical method to illustrate the relationship between ROC curves and decision values. Although ROC curves can be built entirely from ranks, it is possible to plot the decision values versus the false positive rate which creates much more insight into the nature of the classification problem and the performance of the classifier that the ROC curve represents.

1.5.3 Intelligent Selection of Subspaces

One of the hypotheses states that synergistic combination will not occur unless there is intelligent subspace selection. Building models for the intelligent selection of subspaces is an exploratory effort. Three different methods to intelligently select subspaces have been explored. The first method involves selection of orthogonal subspaces through the analysis of subspace principal components. Given a selection of subspaces, it is possible to calculate the principal components of each subspace.

If the principal components of subspaces do not demonstrate correlation, these subspaces are viewed as orthogonal. Uncorrelated principal components indicates that the subspaces are measuring different behavior. The method utilized to find these orthogonal subspaces is evolutionary optimization.

Another technique that is closely related involves the analysis of nonparametric statistics. Once again given a selection of subspaces, the orthogonality of these subspaces can be measured with the use of Kendall's W, a nonparametric statistic that measures concordance (agreement) of rankings. Given a centroid for each subspace, the distance of an instance can be measured to the centroid. For each subspace, given instances $1...n$, each instance can be ranked based on the distance to the subspace centroid. The correlation of the ranked distances is calculated with Kendall's W. We seek a low value for Kendall's W, indicating discordance; this is again pursued with evolutionary optimization.

A third subspace modeling effort exploits the covariance structure of the data. Experiments have been performed with subspaces constructed from hierarchical clustering of the variables based upon their covariance as a similarity measure. Experiments have also been performed with subspaces constructed with maximally covariant variables in subspace one, minimally covariant variables in subspace 3, and the remaining variables in subspace 2 (as an example for a model with number of subspaces = 3). Each of these alternatives for intelligent subspace modeling is an innovative approach, however rigorous testing of the effectiveness of these subspace modeling techniques is ongoing.

1.5.4 The Fuzzy ROC Curve and Synergistic Classifier Fusion

Given several subspace models, it is necessary to fuse the decision results to determine the ultimate prediction. This fusion occurs with the application of a technique inspired by fuzzy logic. The prediction models considered in this research create a soft-valued decision value for each instance. Let us refer to this decision value as $d \in \mathbb{R}^1$. For the i^{th} observation predicted by the j^{th} subspace model, the soft label will be d_{ij}. d_{ij} is not a probability or a known distribution; it simply indicates the relative confidence of a model that the instance belongs to one class

or the other. The fusion of d_{ij} across several subspace models requires some type of scaling enabling comparative analysis. Scaling this value as a simple rank has shown exceptional promise for fusion; ranks provide a comparable and meaningful measure for fusion. The rules used to fuse the decision values invoke fuzzy logic. These ranks are then mapped to range in (0,1) without the loss of any information, and T-norm and T-conorm aggregation operators provide a method for fusion [83]. The fusion method of choice will reflect the risk aversion of the decision maker, indicating the decision makers aversion to either false positives or false negatives. Furthermore, rather than blindly fusing all decisions with the same aggregator, the model also addresses methods for fusing contentious, or disagreeing decisions. It is often appropriate to fuse contentious decisions with a different aggregation rule.

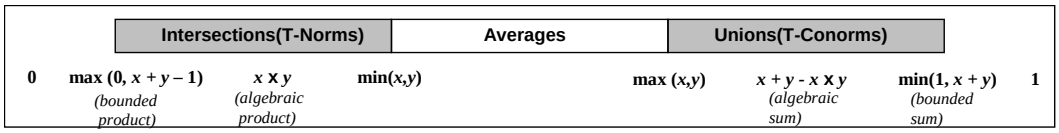

Figure 1.2: Fuzzy aggregation operators

The measure of effectiveness for subspace modeling and fusion involves comparing performance of these intelligent subspace modeling techniques against both the base model and fusion of randomly selected subspaces. The performance of the novelty detection method can be benchmarked by running the method with every variable in one subspace. This will be referred to as the base model. This base model is often a poor performer due to degradation created by a high-dimensional, noisy input space.

It is not uncommon for randomly selected subspaces that are modeled and fused to consistently perform better than this base model. The effectiveness of alternate aggregation operators can be explored with these randomly selected subspaces. The effectiveness of the subspace modeling is measured by comparing performance of randomly selected subspaces and a fixed fusion method versus the intelligently selected subspaces. Results of intelligent subspace modeling and fuzzy ROC curve fusion has been reported in [48, 49].

Promising results from the fusion of ranks produced by prediction models led to the analysis of rank distributions and pseudo ROC curves. This analysis spawned

some of the most important results in this research. Class balance, a parameter of every prediction model that is rarely considered and almost never included as part of the model, provides crucial insight into the behavior of prediction models. If it is possible to assume the balance of the classes in a binary classification problem, it is possible to use fuzzy logic aggregators to create synergistic and improved prediction. Chapter 6 details these results.

1.5.5 Contributions to Computer Intrusion Detection

The primary empirical vehicle and an immediate application for this research is the computer intrusion detection problem. Computer intrusion detection is a broad field in itself, and in general terms it can be considered an unbalanced classification problem. There are two types of intrusion detection: Host-Based Intrusion Detection Systems (HIDS) and Network-Based Intrusion Detection Systems (NIDS). Host-based systems monitor the behavior of workstations and user profiles. HIDS typically detect anomalies. Results published in [50] illustrate datamining and classification techniques for a host-based intrusion detection problem that involved identifying authentic users versus intruders based upon command usage history of users. This same type of model could be applied to user logs that exist on every computer. NIDS monitor network behavior and typically detect attacks with signature-based models (identify signatures of known attacks) and / or anomalies. In order to model both host based data and network based data with the machine learning techniques utilized in this research, numeric data must be extracted. The variables created to represent the data span the full spectrum of creativity; some variables are simple rates and averages, other variables are products of interesting data mining routines. The data collection model and representation of the host and network data is a byproduct of this research, however it is a contribution to the computer intrusion detection community that should not be overlooked.

1.5.6 The Curse of Dimensionality, Kernels, and Class Imbalance

The curse of dimensionality is a commonly known problem in datamining and machine learning. Machine learning and data mining typically seek to show a degree of similarity between observations. This degree of similarity can be measured

by numerous metrics. Most of these metrics consider some type of distance. A problem with distance metrics in high-dimensional spaces is that distance is typically measured across volume. Volume increases exponentially as dimensionality increases, and points tend to become equidistant. The curse of dimensionality is explained with several artificial data problems in [93]. A simple example of this involves a cube with sides of unit length in hyperspace. If two points within this hypersphere are randomly selected, the variance of the distance between these two points approaches zero as dimensionality increases. If a cube has a side of length that is less than one, the volume of this cube approaches zero with increased dimensionality; if the side is of length greater than one, the volume approaches infinity as dimensionality increases.

Kernel-based pattern recognition, especially in the unsupervised domain, is not entirely robust in high- dimensional input spaces. A kernel is nothing more than a similarity measure between two observations. Given two observations, $\mathbf{x}_1, \mathbf{x_2}$, the kernel between these two points is represented as $\kappa(\mathbf{x}_1, \mathbf{x_2})$. A large value for $\kappa(\mathbf{x}_1, \mathbf{x_2})$ indicates similar points, where smaller values indicate dissimilar points. Typical kernels include the linear kernel, $\kappa(\mathbf{x}_1, \mathbf{x_2}) = \langle \mathbf{x}_1, \mathbf{x_2} \rangle$, the polynomial kernel, $\kappa(\mathbf{x}_1, \mathbf{x_2}) = (\langle \mathbf{x}_1, \mathbf{x_2} \rangle + 1)^p$, and the popular gaussian kernel, $\kappa(\mathbf{x}_1, \mathbf{x_2}) = e^{(-\|\mathbf{x}_1 - \mathbf{x_2}\|/2\sigma^2)}$.[1] As shown, these kernels are all functions of inner products. If the variables within $\mathbf{x}_1$ and $\mathbf{x_2}$ are considered random variables, these kernels can be modeled as functions of random variables. The fundamental premise of pattern recognition is that $(\kappa(\mathbf{x}_1, \mathbf{x_2})|y_1 = y_2) > (\kappa(\mathbf{x}_1, \mathbf{x_2})|y_1 \neq y_2)$. If this premise is consistently true, good performance occurs. By modeling these kernels as functions of random variables, it can be shown that the addition of noisy, meaningless input variables degrades performance. This noise increases the variance of the kernels, and increased variance degrades the likelihood of the fundamental premise shown above.

For balanced supervised classification problems, feature selection methods exist for eliminating meaningless noisy variables. This is not the case in the security problem. Certain properties may indicate a noisy variable, but it is not possible to

[1] Throughout this text the inner product, or dot product betwee two vectors will be represented as $\langle \cdot, \cdot \rangle$.

determine whether or not the variable is meaningless. The notion of modeling in subspaces is to dilute the impact of the noise while retaining every variable.

In a classification problem, the curse of dimensionality is a function of the degree of imbalance. If there are a small number of positive examples to learn from, feature selection is possible but difficult and the evidence required to illustrate that a feature is not meaningful is burdensome. If the problem is balanced, the burden is not as great. Features are much more easily filtered and selected.

A simple explanation of this is to consider a two-sample Kolmogorov test [123]. This is a classical statistical test to determine whether or not two samples come from the same distribution, and this test is general regardless of the distribution. This indicates that the test can be performed without knowing the underlying distribution of the data. In classification models, a meaningful variable should behave differently depending on the class, implying distributions that are not equal. Stated in terms of distributions, if x is any variable taken from the space of all variables in the dataset, $(F_x(x)|y = 1)$ should not be equivalent to $(G_x(x)|y = -1)$. $F_x(x)$ and $G_x(x)$ simply represent the cumulative distribution functions of $(x|y = 1)$ and $(x|y = -1)$, respectively. In order to apply the two-sample Kolmogorov test, the empirical distribution functions of $F_x(x)$ and $G_x(x)$ must be calculated from a given sample, and these distribution functions will be denoted as $F^*_{N_1}(x)$ and $G^*_{N_2}(x)$. N_1 will equate to the number of samples in the minority class, and N_2 equates to the number of samples in the majority class. These empirical distribution functions are easily derived from the order statistics of the given sample, which is shown in [123]. The Kolmogorov two-sample test states that if the supremum of the difference of these functions exceeds a tabled critical value depending on the modeler's choice of α(sum of probabilities in two tails), then these two distributions are significantly different. Stated formally, our hypothesis is that $F_x(x) = G_x(x)$. We reject this hypothesis with a confidence of $(1 - \alpha)$ if equation 1.1 is true.

$$D_{N_1,N_2} = \sup_{-\infty<x<\infty} |F^*_{N_1}(x) - G^*_{N_2}(x)| > D_{N_1,N_2,\alpha} \tag{1.1}$$

For larger values of N_1 and N_2 (both N_1 and N_2 greater than 20) and $\alpha = .05$, we can consider equation 1.2 to illustrate an example. This equation is found in the

tables listed in [123]:

$$D_{N_1,N_2,\alpha=.05} = 1.36\sqrt{\frac{N_1+N_2}{N_1 N_2}} \tag{1.2}$$

If N_2 is fixed at 100, and N_1 is considered the minority class, it is possible to plot the relationship between N_1 and the critical value necessary to reject the hypothesis.

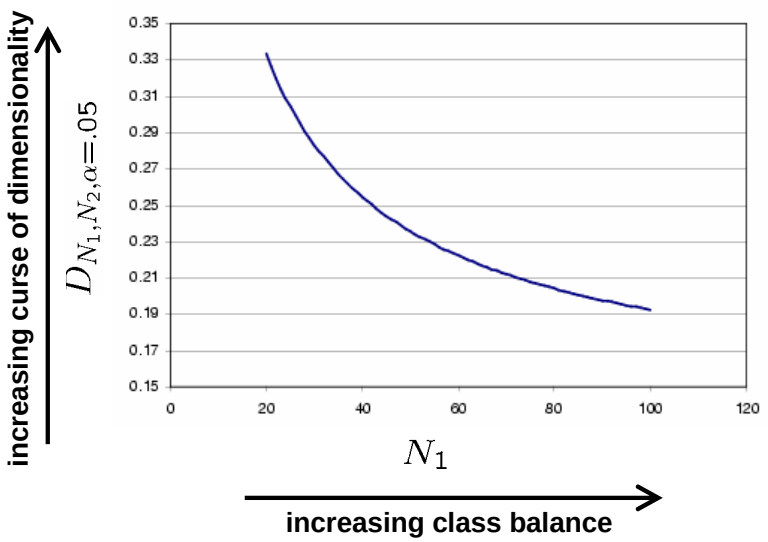

Figure 1.3: Plot of critical value for two-sample Kolmogorov test with fixed N_2, $\alpha = .05$

Figure 1.3 illustrates the effect of class imbalance on feature selection. If the classes are not balanced, as is the case when $N_1 = 20$ and $N_2 = 100$, there is a large value required for D_{N_1,N_2}. It is also evident that if the classes were more severely imbalanced, D_{N_1,N_2} would continue to grow exponentially. As the classes balance, D_{N_1,N_2} and the critical value begins to approach a limit. The point of this exercise was to show that the curse of dimensionality is a function of the level of imbalance between the classes, and the two sample Kolmogorov test provides a compact and statistically grounded explanation for this.

1.6 Kernel Behavior in High-Dimensional Input Space

The previous subsections list the innovative claims of this research. Most of these innovative claims rely on the assumption that noisy high-dimensional input space can degrade the performance of kernel methods, especially given a security classification problems. In order to further support this assumption, an example is given in this section which illustrates the impact of dimensionality on linear kernels and gaussian kernels.

Consider two random vectors that will serve as artificial data for this example.

$$\mathbf{x_1} = (z_1, z_2, ..., z_m),\ z_i \sim N(0,1) \text{ i.i.d}$$

$$\mathbf{x_2} = (z_{1'}, z_{2'}, ..., z_{m'}),\ z_{i'} \sim N(0,1) \text{ i.i.d}$$

$m' = m$, and let $v_i = z_i z_{i'}$

The expected value of v_i is zero. v_i is the product of two standard normal random variables, which follows an interesting distribution discussed in [65]. The plot of this distribution is shown in figure 1.4.

To find the expectation of a linear kernel, it is straightforward to see that $E(\langle \mathbf{x}, \mathbf{y} \rangle) = \sum_i v_i = E(z_1 z_{1'} + z_2 z_{2'} + ... + z_m z_{m'}) = 0$. The variance of the linear kernel can be found as follows:

$$f_{z_i, z_{i'}}(z_i, z_{i'}) \text{ is bivariate normal } \Rightarrow f_{z_i, z_{i'}}(z_i, z_{i'}) = \frac{1}{2\pi} e^{\frac{-(z_i^2 + z_{i'}^2)}{2}}$$

$$f_v(v) = \int_{-\infty}^{\infty} f_{z_i, z_{i'}}(z_i, \frac{v}{z_i}) \frac{1}{|z_i|} dz_i$$

$$E(v) = 0 \Rightarrow \text{ variance } = E(v^2) = \int_{-\infty}^{\infty} v^2 [f_v(v)] \partial v = 1$$

(verified by numerical integration)

Again considering the linear kernel as a function of random variables, $\kappa(\mathbf{x_1}, \mathbf{x_2}) = \langle \mathbf{x_1}, \mathbf{x_2} \rangle = \sum_{i=1}^{m} v_i$ is distributed with a mean of 0 and a variance of $\sum_{i=1}^{m} 1 = m$.

In classification problems, however, it is assumed that the distributions of the

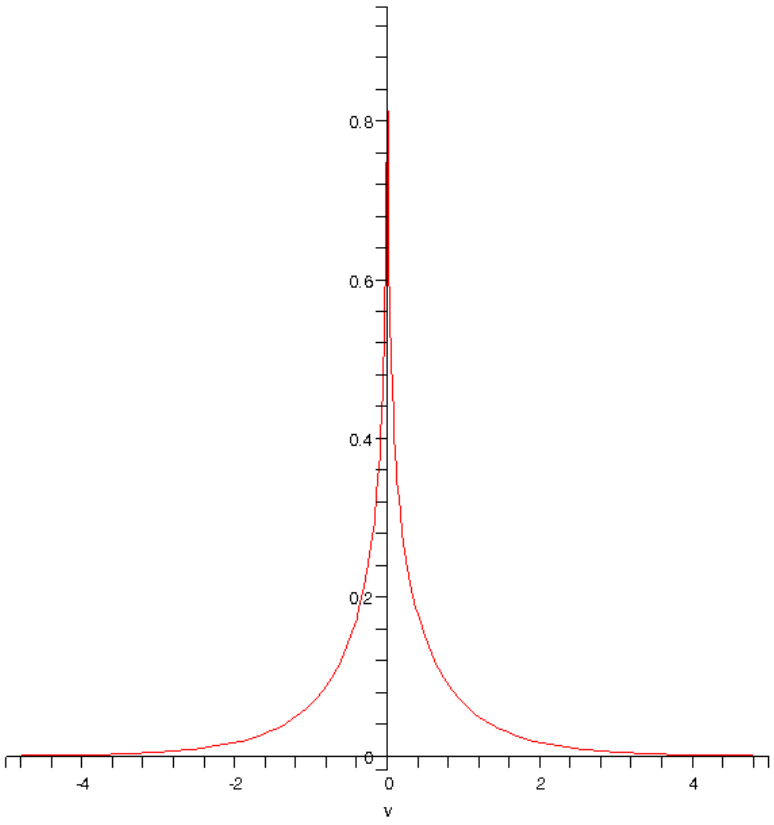

Figure 1.4: Plot of of the product of two standard normal random variables, $v_i = z_i z_{i'}$

variables for one class are not the same as the distributions of the variables for the other class. Let us now consider v_- as a product of dissimilar distributions, and v_+ as a product of similar distributions. Let $v_- = (z_i - 1)(z_{i'} + 1)$. v_- will be distributed with a mean of $\mu_- = E(z_i z_{i'} - z_{i'} + z_i - 1) = -1$, and a variance of 3 (verified through numerical integration). The linear kernel of the dissimilar distributions can be expressed as:

$$\kappa(\mathbf{x_1} - 1, \mathbf{x_2} + 1) = \sum_{i=1}^{m} v_-$$

This linear kernel is distributed with the following parameters:

$$\text{mean}_- = m\mu_- = -m, \text{ variance } = m\sigma^2 = 3m$$

For the similar observations, let $v_+ = (z_i + 1)(z_{i'} + 1) = (z_i - 1)(z_i' - 1)$.

The parameters of the kernel for the similar observations can be found in the same manner. v_+ is distributed with a mean of $\mu_+ = E(z_i z_{i'} + z_{i'} + z_i + 1) = 1$ and a variance of $\sigma^2 = 3$. The linear kernel of the similar distributions can be expressed as:

$$\kappa(\mathbf{x_1} + 1, \mathbf{x_2} + 1) = \sum_{i=1}^{m} v_+$$

This kernel is distributed with the following paramaters:

$$\text{mean}_+ = m\mu_+ = m, \text{ variance } = m\sigma^2 = 3m$$

The means and variances of the distributions of the linear kernels are easily tractable, and this is all the information that we need to analyze the effect of dimensionality on these kernels. In the above example, the mean of every variable for dissimilar observations differs by 2. This is consistent for every variable. Obviously, no dataset is this clean, however there are still interesting observations that can be made. Consider that rather than each variable differing by 2, they differ by some value ϵ_i. If ϵ_i is a small value, or even zero for some instances (which would be the case for pure noise), this variable will contribute minimally in distinguishing similar from dissimilar observations, and furthermore the variance of this variable will be entirely contributed. Also notice that at the rate of $3m$, variance grows large fast.

Based on this observation, an assertion is that for the binary classification problem, bimodal variables are desirable. Each mode will correspond to either the positive or negative class. Large deviations in these modes, with minimal variation within a class, are also desired. An effective model must be able to distinguish v_- from v_+. In order for this to occur, the model needs good separation between $mean_-$ and $mean_+$ and variance that is under control.

It is also interesting to explore the gaussian kernel under the same example. For the gaussian kernel, $\kappa(\mathbf{x_1}, \mathbf{y_2}) = e^{-\|\mathbf{x_1} - \mathbf{x_2}\|^2/2\sigma^2}$. This kernel is entirely dependent upon the behavior of $\| \mathbf{x_1} - \mathbf{x_2} \|^2$ and the modeler's choice of the parameter σ (which has no relation to variance).

Restricting our attention to $\| \mathbf{x_1} - \mathbf{x_2} \|^2$, an initial observation is that this

expression is nothing more than the euclidean distance squared. Also, if $\mathbf{x_1}$ and $\mathbf{x_2}$ contain variables that are distributed $\sim N(0,1)$, then $(\mathbf{x_1} - \mathbf{x_2})$ contains variables distributed normally with a mean of 0 and a variance of 2.

Let $w = (z_i - z_{i'})^2$, implying that $w/2$ is a chi-squared distribution with a mean of one (which will be annotated as $\chi^2(1)$). This also indicates that $w = 2\chi^2(1)$, indicating that w has a mean of 2 and a variance of 8 (verified by numerical integration).

Therefore, $\parallel \mathbf{x_1} - \mathbf{x_2} \parallel^2 = \sum_{i=1}^{m} w_i$ will have a distribution with a mean of $2m$ and a variance of $8m$. Notice that the variance grows much faster under this formulation, indicating even more sensitivity to noisy variables.

The purpose of the above example is to show how every variable added will contribute to the overall behavior of the kernel. If the variable is meaningful, the pattern contributed to the -1 class is not equivalent to the pattern contributed to +1 class. The meaningfulness of the variable can also be considered in terms of cost and benefit. The benefit of including a variable in a classification model is the contribution of the variable towards pushing $mean_-$ away from $mean_+$. The cost of including a variable involves the variance. This variance will be included regardless of the significance of the benefit.

The goal is to include variables where this benefit outweighs the cost. Feature selection manages this behavior for supervised classification models. For security classification models, an alternative method to achieve this goal is to model in subspaces. Subspaces reduce variance and mitigate cost of noisy variables, however subspace modeling is complicated with the task of intelligent selection of subspaces and proper fusion of the subspace models.

1.7 Organization of this Document

Chapter 2 provides a summary of related mathematical models and some of the previous work that relates to the research included in this dissertation. Chapter 3 is an overview on ROC curves, focusing on the construction of ROC curves and including a novel representation of the relationship between ROC curves and decision variables, the *Decision ROC Chart*. Chapter 4 discusses the methods utilized

to model computer intrusion detection problems. This includes techniques to create variables from both host based intrusion detection and network based intrusion detection. Chapter 5 details exploratory methods undertaken to find synergistic classifier ensembles. Much of this chapter discusses efforts for the creation of intelligent subspaces. The chapter details several subspace analysis methods which represent possible avenues for intelligent subspace selection. Chapter 6 includes the most important findings in this dissertation. The chapter introduces pseudo-ROC curves, rank distributions, and the synergistic fusion of ranks to improve classifier performance. Chapter 7 discusses the gaussian kernel matrix and insights regarding the statistical properties of this matrix. These statistical properties provide potential avenues for the automated tuning of the gaussian kernel. Chapter 8 presents three applied cases. Each of these cases represent security classification problems that have been investigated using principles discussed in this research. The appendices include comments on the computing methods and tools utilized as well as a discussion of some text mining techniques utilized to analyze a host based computer intrusion detection problem.

CHAPTER 2
REVIEW OF LITERATURE AND RELATED MATHEMATICAL MODELS

2.1 Introduction

The literature and mathematical models that apply to this research involve many domains. In the interest of the organization of this chapter, each section will discuss the material reviewed from a particular domain. These domains include the related mathematical models, ROC curves (output analysis), and recent work in computer intrusion detection which will cover both host based (masquerading) intrusion detection and network based intrusion detection. Throughout this chapter the Schonlau et. al. dataset is abbreviated as the SEA dataset. This is a host-based intrusion detection dataset that provided initial results for the work with the models discussed. A detailed description of this dataset is included in Chapter 4.

2.2 Selected Mathematical Models for Unbalanced Binary Classification

There are numerous mathematical models used in binary classification problems, and this section introduces several of the techniques explored and applied for this research. Included is a brief introduction to each model or technique and typically a short discussion of known applications or initial experimental results. Many of the models have alternate formulations or perhaps even several algorithms which achieve the same result. Only the algorithms which have been studied and implemented have been included.

Dimension reduction is an important concept that includes several techniques, and there is a brief discussion on this concept and the application to the security problem. Scaling of data will be described as "Mahalanobis" scaling of the data. This simply means that we scale each variable by subtracting from it the variable's mean and dividing by the variable's standard deviation. For the actual prediction

modeling there are two models specifically discussed. The first model is a supervised classification technique. This model is Kernel Partial Least Squares (KPLS) [124]. The model selected for unsupervised classification is the One-Class Support Vector Machine (SVM), which is a variant of the traditional (SVM) [127].

2.2.1 Statistical Dimension Reduction Techniques

Dimension reduction techniques are important statistical tools that are often necessary to create manageable data sets (meaning that rapid computing is possible with the data), explain interaction between variables, indicate root causes, and capture the most important information with many fewer dimensions. Computer intrusion detection, and security problems in general, can involve mountains of data. Introduce any type of multimedia data and file sizes explode. If this multimedia needs to be represented as some quantifiable variable, it is often necessary to employ dimension reduction. Dimension reduction techniques can increase computing speed and reduce storage. Furthermore, these techniques are not destructive to the data, meaning that datasets can be reconstructed from the dimensionally reduced space back to its original space and form. There are a number of dimension reduction techniques to include principal component analysis (PCA), partial least squares (PLS), independent component analysis (ICA), and canonical correlation analysis(CCA). Each of these techniques solves a particular problem, and we will briefly discuss each of these. PLS serves as a regression model, and therefore it will be discussed under the prediction modeling section.

2.2.1.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a very old multivariate statistical analysis technique. If $\mathbf{X}$ is our $N \times m$ data matrix, PCA seeks to solve the equation $\mathbf{S} = \mathbf{XL}$ where we can consider $\mathbf{S}$ as our scores matrix and $\mathbf{L}$ as our matrix of loading vectors. $\mathbf{L}$ is equivalent to $\mathbf{E}$ for the formulation presented here. Harold Hotelling [78] was the pioneer of principal components, and today almost every textbook that serves as an introduction to multivariate statistics will contain a chapter on principal component analysis [85]. PCA is a dimension reduction technique that seeks to find linear combinations of the variables in order to maximize variance along

a single axis. This single axis that contains the maximum variance can be considered the direction of the first principle component. The second principal component captures maximal variance in an orthogonal direction(all principle components are orthogonal to each other and the scores uncorrelated). There can only be as many principal components as there are variables in a data set. Typically, the first few principal components capture most of the variance.

Mathematical preliminaries for PCA include understanding the correlation matrix of the data. There are a few interesting points regarding the correlation matrix. Much confusion exists regarding the difference between the covariance matrix and the correlation matrix. They are not the same. Covariance matrices contain the variance of each variable on the diagonal and the covariance between variables for the off-diagonal elements. The correlation matrix contains a 1 in each diagonal entry and the pairwise correlation between variables in the off diagonal elements. If data are normalized and it is assumed that every variable within the scaled data possesses a mean of 0 and a variance of unity, then the correlation matrix will equal the covariance matrix for this data. Given a $N \times m$ normalized data matrix, $\mathbf{X}$, where every row represents a multivariate observation, the correlation matrix is exactly equal to $(1/(N-1))\mathbf{X^T X}$. The proof for this is quite simple. Consider ρ_{ij} as the correlation between variable i and variable j, σ_{ij}^2 as the unbiased estimator of the covariance between the i^{th} and j^{th} variables, x_{ij} as the i^{th} observation of the j_{th} variable. $\mathbf{x}_j$ is the column vector of all observations for variable j.

$$\rho_{ij} = \frac{\sigma_{ij}^2}{\sigma_{ii}\sigma_{jj}} = \sigma_{ij}^2 \text{ when } \sigma_{ii} = \sigma_{jj} = 1$$

$$\sigma_{ij}^2 = \frac{1}{N-1}\sum_{k=1}^{N}(x_{ki} - \mu_i)(x_{kj} - \mu_j) = \frac{1}{N-1}\sum_{k=1}^{N}(x_{ki})(x_{kj}) \text{ when } \mu_i = \mu_j = 1$$

$$\implies \sigma_{ij}^2 = \rho_{ij} = \frac{1}{N-1}\mathbf{x}_i^T\mathbf{x}_j$$

$$\implies \begin{pmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ \rho_{m1} & \rho_{m2} & \cdots & \rho_{mm} \end{pmatrix} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

A projection of the data on the eigenvectors of the correlation matrix creates the principal components. Consider a matrix of the eigenvectors as $\mathbf{E}$, where every column in the matrix is an eigenvector. We will refer to a diagonal matrix $\mathbf{D}$ that contains the eigenvalues, λ_i, in decreasing order. Every eigenvalue corresponds to a specific eigenvector. The largest eigenvalue corresponds to the eigenvector that represents the first principal component, which captures the most variance. The first eigenvector is the first loading vector.

$$\mathbf{E} = \begin{pmatrix} e_{11} & e_{12} & ... & e_{1m} \\ e_{21} & e_{22} & ... & e_{2m} \\ ... & ... & ... & ... \\ e_{m1} & e_{m2} & ... & e_{mm} \end{pmatrix} \tag{2.1}$$

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 & ... & 0 \\ 0 & \lambda_2 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \lambda_m \end{pmatrix} \tag{2.2}$$

The eigenvectors can be considered as loading vectors, or the coefficients for the linear combinations for each principle component. If we have variables $x_1, x_2, ..., x_m$, we can represent a principal component score as $e_{11}x_1 + e_{12}x_2 + ... + e_{1m}x_m$. These scores can be represented in a plot, and it is often useful to plot the scores of two different principal components on a two dimensional plot for visualization.

Figure 2.1 illustrates both a score plot and a scree plot. The scree plot shows the values for each eigenvalue. A scree plot with a sharp elbow, meaning that the first one or two eigenvalues is large with small successive eigenvalues, indicates that that most of the variance is captured in the first couple of principal components. A gentle sloping scree plot, similar to the one shown, illustrates dispersion of variance

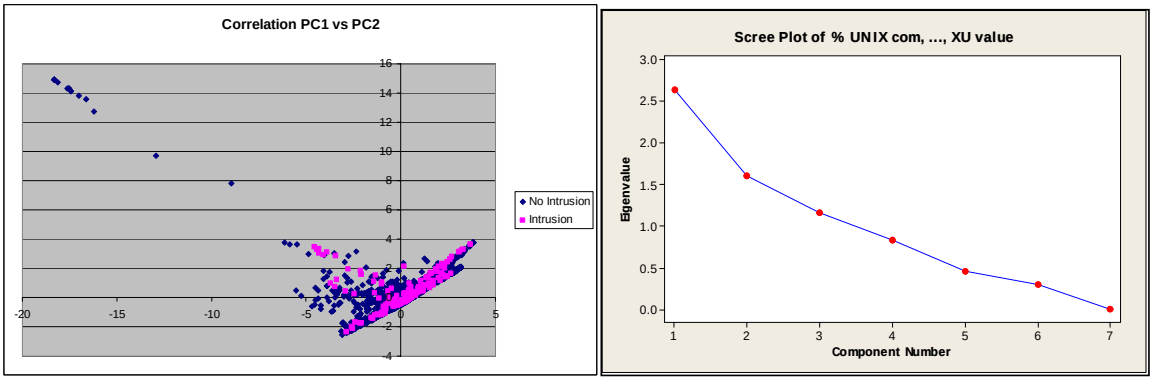

Figure 2.1: Principal component score plot and scree plot

which often indicates a more interesting data set.

In 1966 Herman Wold published a paper that included the Non Iterative Partial Least Squares Algorithm (NIPALS) [146], a technique for calculating the principal components without the cost incurred by directly calculating eigenvectors and eigenvalues (which requires solving a system of linear equations). This algorithm is remarkably elegant and compact, and the style of this algorithm extends to the technique used to extract partial least squares components and independent components. It is worth illustrating this algorithm here so the reader can draw comparisons to the algorithms shown for PLS and ICA component extraction. It is assumed that all data are scaled to a mean of 0 and variance of unity (Mahalanobis scaling).

Algorithm 1 Wold's Non Iterative Partial Least Squares Algorithm

1: **for** $i = 1$ to m **do**
2: Randomly estimate a normalized (length of unity) loading vector, $\mathbf{t}^T$
3: **while** $\mathbf{t} \leftarrow \frac{\mathbf{l}}{\|\mathbf{l}\|}$ has not converged **do**
4: Compute the score vector, $\mathbf{s}=\mathbf{X}\mathbf{t}$
5: Compute new loadings, $\mathbf{l}^T=\mathbf{s}^T\mathbf{X}$, $\mathbf{t} \leftarrow \frac{\mathbf{l}}{\|\mathbf{l}\|}$
6: **end while**
7: $\mathbf{X}=\mathbf{X}\text{-}\mathbf{s}\mathbf{t}^T$
8: $\mathbf{t} = i^{th}$ principal component loading vector
9: **end for**

2.2.1.2 Canonical Correlation Analysis

Canonical correlation analysis is multivariate statistical technique designed for comparing two sets of variables. Conceptualized by Hotelling in 1936 [78], canonical correlation analysis actually measures the correlation that exists between two sets of variables. Consider two sets of variables, $\mathbf{x} = x_1, ..., x_a$ and $\mathbf{y} = y_1, ..., y_b$ that measure a certain phenomena. There are $a + b$ variables. For each observation, one can observe $\mathbf{x}$ and $\mathbf{y}$. Canonical correlation analysis provides a measure of correlation between $\mathbf{x}$ and $\mathbf{y}$ by creating linear combination of each. Consider a linear combination of $\mathbf{x}$, $U_i(\mathbf{x})$, and a similar linear combination of $\mathbf{y}$, $V_j(\mathbf{y})$, such that the correlation between $U_i(\mathbf{x})$ and $V_i(\mathbf{y})$ is maximized when $i = j$, subject to the constraint that the correlation between $U_i(\mathbf{x})$ and $V_j(\mathbf{y})$ is 0 when $i \neq j$. It is additionally stipulated that the variance of $U_i(\mathbf{x})$is equal to the variance of $V_j(\mathbf{y})$ which equals 1. Let us call the correlation that exists between $U_1(\mathbf{x})$ and $V_1(\mathbf{y})$ the first canonical correlation. When $i = j = a$, this is the a^{th} canonical correlation. Several experiments involving the SEA data indicate that canonical correlation between sets of variables is often related to the amount of synergy that exists when these variable sets combine in a multiple classification system. For a more detailed explanation of canonical correlation analysis, see [85, 130].

2.2.1.3 Independent Component Analysis

Independent Component Analysis(ICA) is the newest of all of the dimension reduction techniques. ICA was considered as a potential dimension reduction technique to apply to the security classification problem, however the properties of this technique rendered ICA infeasible for SCPs. Although dimension reduction is possible with ICA, unlike PCA there is no importance or value associated with the components. Any given component could contain noise or critical information for the problem. Worse yet, every time the ICA algorithm solves for the components, ordering of the components can differ. Regardless, ICA is a fascinating new dimension reduction and signal detection technique. The following ICA survey has been included to illustrate the basic nature of this method, its relation to solving SCPs, and perhaps provide enough insight to spur future research towards the application

of ICA to solve SCPs.

ICA appeared in the 1980s and 1990s, however the first comprehensive text written on this subject is written by Aapo Hyvarinen, Juha Karhunen, and Erkki Oja [81]. Hyvarinen also wrote a comprehensive survey paper on ICA [80]. Independent component analysis seeks to find linear combinations of the data which separates the data into independent variables. Again consider our data matrix $\mathbf{X}$ which will be referred to as the mixed signal. We are interested in extracting $\mathbf{S}$, the independent signals. The problem formulation is $\mathbf{X}$=$\mathbf{AS}$ such that $\mathbf{A}$ can be considered a mixing matrix. In [69], the authors illustrate the use of ICA to separate randomly mixed images. ICA is a very useful technique anytime the problem involves a noisy signal such as visual, audio, radar, sonar, and it is necessary to decipher the true signal. A common explanation of independent component analysis is the cocktail party problem. Imagine three conversations that take place at a cocktail party. Three microphones are positioned in the room to record these conversations, all of which can be recorded but each conversation creates conflicting sound and the end result is what sounds like a noisy room. In [81], this problem is formulated in the following manner:

The audio collected on each microphone is x_1, x_2, and x_3, and our three signals are s_1, s_2, and s_3. In order to separate theses signals, a matrix $\mathbf{W}$ should be found such that $\mathbf{S}$=$\mathbf{WX}$. There are several key points about ICA that must be understood before showing the algorithm.

The first step in ICA involves "whitening" the observed data. Whitening consists of transforming the scaled variables into uncorrelated variables with a variance of unity [81]. Whitening can be explained as follows: Given Mahalanobis scaled data in $\mathbf{X}$, determine the matrix $\mathbf{V}$ such that $\mathbf{Z} = \mathbf{VX}$ is the whitened data associated with $\mathbf{X}$. Given the aforementioned definition of whitening, the theory of PCA comes into practice where $\mathbf{V} = \mathbf{D}^{-1/2}\mathbf{E}^T$ where $\mathbf{E}$ and $\mathbf{D}$ are the eigenvector and eigenvalue matrices as discussed in the section on principal components. From this definition, a whitened Z is computed. However, caution is advised when calculating $\mathbf{D}^{-1/2}$. If there are small eigenvalues, which is common, a ridge matrix must be added to $\mathbf{D}$ such that $\mathbf{D}' = \mathbf{R} + \mathbf{D}$, where $\mathbf{R}$ contains a small ridge offset value in the diagonal

and zeros elsewhere.

Whitening is only the first part of ICA. This whitening, or sphering of the data, creates uncorrelated but highly gaussian variables in $\mathbf{Z}$ [81]. In order to eliminate this gaussian property, Hyvarinen discusses the idea of entropy and negentropy. Entropy is used in the context of information theory and statistics, where the more random a variable, the higher the entropy. Entropy is different from variance given that variance measures the dispersion of a probability distribution. Entropy is an overall index that illustrates the degree of equivalence of the cumulative distribution function across the entire distribution, or the randomness of the variable. Entropy, measured by the function $H(y)$, can be estimated for the observed data using a technique that estimates y as a gaussian function compensated by the nongaussian properties of y through a Taylor series expansion of the Kurtosis functions of y [81]. There are several numerical computing techniques that can be used to estimate negentropy. Now the idea is to extract independent components by maximizing negentropy.

Hyvarinen created the following algorithm, known as FastICA. This is another iterative algorithm that uses deflation to extract orthogonal vectors. The similarities that exist between NIPALS for PCA, NIPALS for PLS, and FastICA are remarkable. Here is the algorithm from [81]:

Algorithm 2 Hyvarinen's FastICA Algorithm

1: Choose m, the number of ICs to estimate
2: **for** $p = 1$ to m **do**
3: Randomly estimate a normalized (length of unity) vector $\mathbf{t}$
4: **while** $\mathbf{t} \leftarrow \frac{\mathbf{w}_p}{\|\mathbf{w}_p\|}$ has not converged **do**
5: $\mathbf{w}_p = \mathbf{t}$
6: $\mathbf{w}_p = E\{\mathbf{z}g(\mathbf{w}_p^T\mathbf{z})\}$ - $E\{g'(\mathbf{w}_p^T\mathbf{z})\}\mathbf{w}$
7: $\mathbf{t} \leftarrow \frac{\mathbf{w}_p}{\|\mathbf{w}_p\|}$
8: **end while**
9: **end for**

Initial results with this algorithm applied to IDS data are promising. Natural clusters occur often with independent components, and this is particularly useful for unsupervised learning problems. Similar to the pursuit of using principal components as features, independent components can create features as well. There is

innovative opportunity involving the use of independent components as features, to include a thorough analysis of how to create these features and their impact in predictive modeling. Independent components have a natural tendency to filter noise and present underlying independent patterns. Noise is exactly what we are trying to eliminate in the security classification problem. The following example with images provides a good example of how independent components filter this noise.

Using an implementation of the FastICA algorithm implemented in MATLAB, two personal photos are randomly mixed with noise to create a source separation problem. This example uses four images as the initial source vectors (the vectors are the pixel values for the grey-scale image, an integer between 0 and 256), creating $\mathbf{S}$. These four images are mixed with a mixing matrix, $\mathbf{A} \in \mathbb{R}^{4\times4}$. The first image is two children in an office, the second image is a soldier wearing a ski cap, the third image is random noise, and the fourth image is a meaningless pattern.

The mixing matrix, $\mathbf{A}$, is considered unknown and not utilized at all in the algorithm to separate the pictures. An important subtlety of this source separation included proper scaling. Independent components are completely oblivious to scale. The solution is to Mahalanobis scale the results, and then project this back into the desired scale.

Independent components are unlike principal components with regard to order and relation to eigenvalues. Given the same ICA problem, two different iterations will uncover the source signals but they often will not occur in the same order. Therefore, every independent component is just as significant as the other, unlike PCA. Additionally, the sign of independent components are ambiguous. The picture of the children is inverted in the results, resulting in an image that looks like a photo negative.

2.2.2 Taxonomy of the Security Classification Problem

In an effort to improve understanding of the types of problems and models available within the domain of security classification, a brief taxonomy is presented. This taxonomy is important because it clarifies terms used throughout machine learning which are often wrongly exchanged and misunderstood, and secondly it

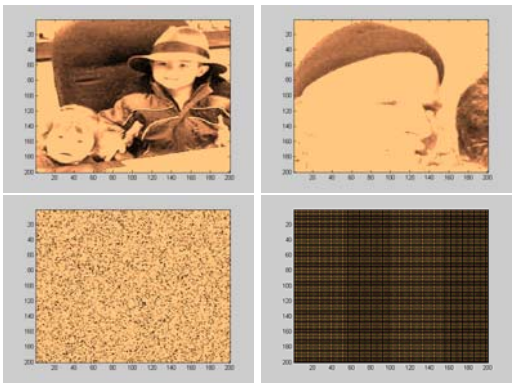

Figure 2.2: Initial images or S

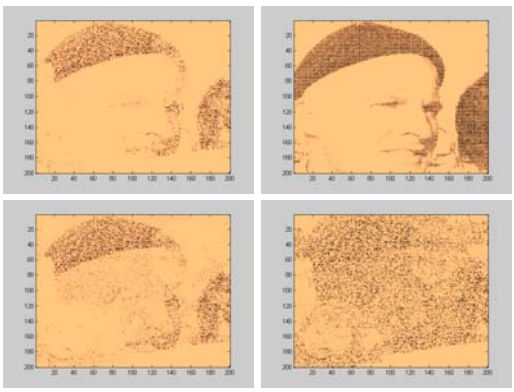

Figure 2.3: Mixed images or X = AS

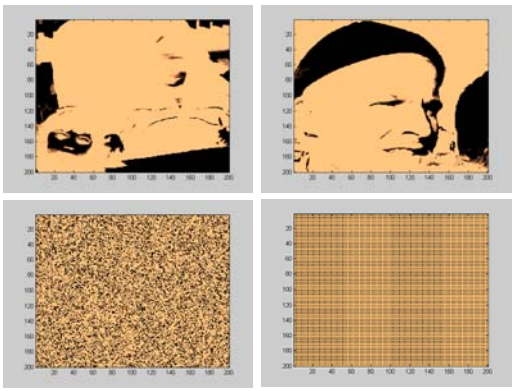

Figure 2.4: Results of ICA attempting to uncover S

provides examples of the types of models that could be used depending upon the problem.

Recall the definition of the security classification problem: a high-dimensional unbalanced binary classification problem which could exist in either the supervised and unsupervised domain. The security classification problem is often unsupervised. If the classes of the data are either unknown or all negative, this is an unsupervised problem. There are three types of unsupervised problems identified in this taxonomy.

1. Novelty Detection. The one-class SVM is a good example of a novelty detection algorithm. This type of algorithm trains on negative data, and based upon the behavior of negative data, attempts to predict the instances of positive data in the test set.

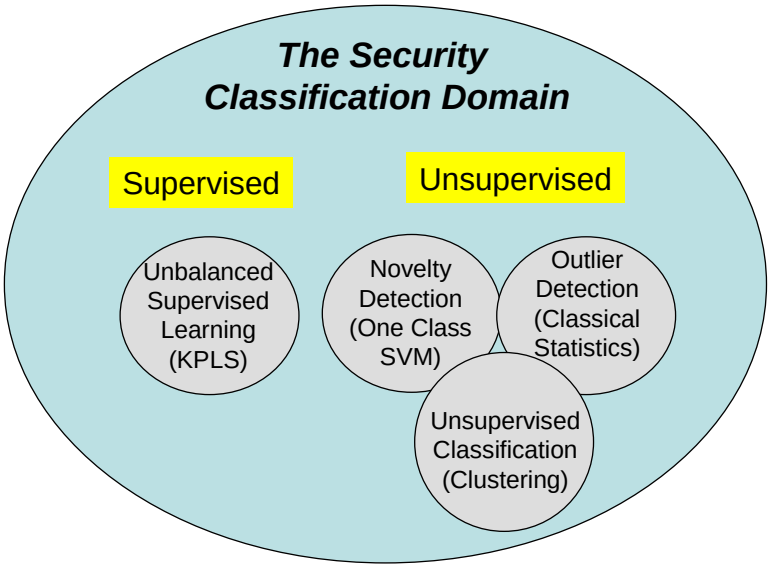

Figure 2.5: Taxonomy of problems in the security classification domain.

2. Unsupervised Classification (Clustering). In clustering, the classes are completely unknown and there typically is no training. Clustering utilizes similarity metrics to cluster similar instances. Clustering algorithms exist for both an unknown number of classes and a predefined number of classes. K nearest neighbor, K means, Hierarchical, and Density Based are several types of clustering algorithms.

3. Outlier Detection. In classical statistics, outliers are a common term. Given a sample of data, instances which deviate a specified measure from the mean are labeled as outliers. An outlier detection approach could exist without any training of the data, or if training occurs with only healthy data, this would be identical to novelty detection.

Supervised learning is also a critical component of the security classification problem. The critical component needed to apply supervised learning is instances of the positive class. Typically these instances are few, therefore making feature selection somewhat difficult. One of the best suited supervised algorithm for this problem is Kernel Partial Least Squares, because it performs a natural feature selection method and responds well to a minority class by scaling of the response.

2.2.3 Prediction Models

There are two prediction models that have been applied to the host-based computer intrusion detection data and the network intrusion detection data. For supervised modeling, Kernel Partial Least Squares (K-PLS) is a powerful technique that will be discussed. If unsupervised modeling is necessary, the model of choice is the one-class support vector machine (SVM).

2.2.3.1 Kernel Partial Least Squares (K-PLS)

Partial Least Squares Regression (PLS) was conceived by the Swedish statistician Herman Wold for econometrics modeling of multi-variate time series [146]. The first PLS publication was a sociology application in 1975 [147]. His son, Svante Wold, applied PLS to chemometrics in the early eighties [148,149] and currently PLS has become one of the most popular and powerful tools in chemometrics, mainly because of the quality of building models with many variables. PLS is not easy to explain and the mathematics involved are far from transparent. Partially for that reason PLS has a low emphasis in mainstream statistics and machine learning.

K-PLS is a technique that has grown from partial least squares analysis. The study of partial least squares(PLS) regression is similar to principal components regression (PCR). Foundations for principal components analysis have been presented in section 2.2.1.1.

PLS analysis considers the response vector (or matrix for multiple responses), typically denoted as $\mathbf{Y}$. PLS regression is a technique that maximizes latent variable correlation with a response vector. Therefore, the first latent variable (which is again a linear combination of the input variables), possesses maximum correlation with the response variable while remaining orthogonal to the remaining latent variables. Since the first few partial least squares components or latent variables capture the majority of correlation with the response variable, powerful prediction models result with desirable dimension reduction properties.

Rosipal explains in [124] how to extract these PLS components. Utilizing the NIPALS approach, algorithm 3 illustrates Rosipal's method.

At each full iteration (completion of step 10), we will store the **t,u,w** and **c**

Algorithm 3 PLS extraction with NIPALS

1: **for** $i = 1$ to m **do**
2: randomly initialize $\mathbf{u}$
3: **while** $\mathbf{u}$ and $\mathbf{t}$ have not converged **do**
4: $\mathbf{w} = \mathbf{X}^T\mathbf{u}$
5: $\mathbf{t} = \mathbf{X}\mathbf{w}$, $\mathbf{t} \leftarrow \frac{\mathbf{t}}{\|\mathbf{t}\|}$
6: $\mathbf{c} = \mathbf{Y}^T\mathbf{t}$
7: $\mathbf{u} = \mathbf{Y}\mathbf{c}$, $\mathbf{u} \leftarrow \frac{\mathbf{u}}{\|\mathbf{u}\|}$
8: **end while**
9: deflate $\mathbf{X}$,$\mathbf{Y}$:
$\mathbf{X} \leftarrow \mathbf{X} - \mathbf{t}\mathbf{t}^T\mathbf{X}$
$\mathbf{Y} \leftarrow \mathbf{Y} - \mathbf{t}\mathbf{t}^T\mathbf{Y}$
10: Assign $\mathbf{t}$ as the i^{th} PLS component
11: **end for**

vectors. These vectors will create matrices $\mathbf{T}$,$\mathbf{U}$,$\mathbf{W}$ and $\mathbf{C}$ which will be used to complete the PLS regression model. If we write the typical regression model as

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{F}$$

where $\mathbf{B}$ is our regression matrix and $\mathbf{F}$ is the residual matrix, Rosipal shows in [124] the following:

$$\mathbf{B} = \mathbf{X}^T\mathbf{U}(\mathbf{T}^T\mathbf{X}\mathbf{X}^T\mathbf{U})^{-1}\mathbf{T}^T\mathbf{Y}$$

The key to Kernel PLS (KPLS) is realizing the kernel matrix formed in the algorithm shown above between steps 2 and 3. The algorithm for KPLS is no different than what is shown for the PLS algorithm, except steps 2 and 3 combine to create:

$$\mathbf{t} = \phi\phi^T\mathbf{u},\ \mathbf{t} \leftarrow \frac{\mathbf{t}}{\|\mathbf{t}\|}$$

ϕ represents the nonlinear function, or kernel function, that transforms the input variables into feature space. $\phi\phi^T$ is the well known kernel matrix. KPLS provides the attractive aspect of feature reduction while also combining the powerful similarity technique that exists within nonlinear kernels. KPLS has been extensively benchmarked with other models such as support vector machines and often yields nearly identical results [9].

KPLS was the original model applied to the SEA dataset for this research.

2.2.3.2 The One-Class Support Vector Machine (SVM)

The one-class SVM is an unsupervised learning technique, sometimes referred to as outlier detection or anomaly detection, originally proposed in [137]. The purpose of a one-class SVM is to learn the attributes of known, "healthy" cases without ever learning from an "unhealthy" or positive case. When introduced to a test set that contains both healthy and unhealthy, or intruders and non-intruders, the one-class SVM utilizes what it learned from the healthy cases to segregate the classes.

The software package utilized to implement the one-class SVM is called LIBSVM (library of support vector machines) [24]. This software employs both the one-class SVM and the more popular supervised learning version of the SVM. The programs are well written in C++ and C, and there is thorough documentation for both understanding the software and understanding the underlying theory. The source code is available, and this software is widely cited and utilized in machine learning applications.

The one-class SVM is a less popular but natural extension from the supervised SVM which is has become a very popular learning technique. Cristianini and Shawe-Taylor provide a detailed explanation of one-class SVMs in [130]. Stolfo and Wang [132] successfully apply the one-class SVM to the SEA dataset and compare it with several of the techniques mentioned above. Chen uses the one-class SVM for image retrieval [27]. The simplest way to express the one-class SVM is to envision a sphere or ball, and the object is to squeeze all of the training data into the tightest ball feasible. This is analogous to the idea of variance reduction for distribution estimation; given a set of data, we want to estimate a distribution that tightly defines this data, and any other data that does not look the same will not fit this distribution. In other words, once the distribution is estimated, data that does not fit the distribution will be considered an outlier or not a member of the negative class. Consider the following formulation of the one-class SVM originally from [127] and also clearly explained in [27]:

Due to the applicability of the one-class SVM to the security classification problem and the innovative research opportunities involving potential enhancement

of this learning model, a detailed derivation of this technique follows. Consider $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l \in \mathcal{X}$ instances of training observations, and Φ is a mapping into the feature space, F, from $\mathcal{X}$.

The following minimization function attempts to squeeze R, which can be thought of as the radius of a hypersphere, as small as possible in order to fit all of the training samples. If a training sample will not fit, ζ_i is a slack variable to allow for this. A free parameter, v, enables the modeler to adjust the impact of the slack variables.

$$\min_{R\in\mathbb{R},\zeta\in\mathbb{R}^l,c\in F} \quad R^2 + \frac{1}{vl}\sum_i \zeta_i \tag{2.3}$$

$$\text{subject to} \quad \| \Phi(\mathbf{x}_i) - c \|^2 \leq R^2 + \zeta_i, \quad \zeta_i \geq 0 \text{ for } i \in [l]$$

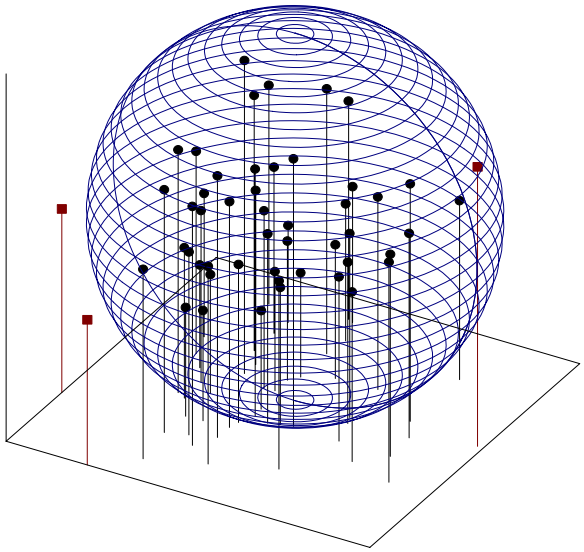

Figure 2.6: A three dimensional visualization of an enclosing sphere. Several novelties are shown as squares.

In order to solve this optimization problem, the following steps occur:

1. Place constraints in standard form:

$$\text{subject to} \quad \| \Phi(\mathbf{x}_i) - c \|^2 - R^2 - \zeta_i \leq 0, \quad -\zeta_i \leq 0 \text{ for } i \in [l]$$

2. Find max of Lagrangian:

$$\mathcal{L}(R^2, c, \zeta_i) = R^2 + \frac{1}{vl}\sum_i \zeta_i + \sum_i \alpha_i(\parallel \Phi(\mathbf{x}_i) - c \parallel^2 - R^2 - \zeta_i) - \sum_i \beta_i \zeta_i$$

$\frac{\partial \mathcal{L}}{\partial R} = 2R - 2R\sum_i \alpha_i = 0 \Rightarrow \sum_i \alpha_i = 1$
$\frac{\partial \mathcal{L}}{\partial c} = 2\sum_i \alpha_i(\Phi(\mathbf{x}_i) - c) = 0 \Rightarrow \sum_i \alpha_i \Phi(\mathbf{x}_i) - c \Rightarrow \sum_i \alpha_i \Phi(\mathbf{x}_i) = c$
$\frac{\partial \mathcal{L}}{\partial \zeta_i} = \frac{1}{vl} - \alpha_i - \beta_i = 0$
R^2 cancel $\Rightarrow$
$\mathcal{L}(R^2, c, \zeta_i) = \cancel{\frac{1}{vl}\sum_i \zeta_i} - \cancel{\sum_i \zeta_i \alpha_i} - \cancel{\sum_i (\frac{1}{vl} - \alpha_i)\zeta_i} + \sum_i \alpha_i \parallel \Phi(\mathbf{x}_i) - c \parallel^2$

3. The following algebraic expansion leads to equation 7.3:

$\parallel \Phi(\mathbf{x}_i) - c \parallel^2 = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_i) \rangle - 2\langle \Phi(\mathbf{x}_i), c \rangle + \langle c, c \rangle$
$= \kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\langle \Phi(\mathbf{x}_i), \sum_j \alpha_j \Phi(\mathbf{x}_j) \rangle + \parallel \sum_j \alpha_j \Phi(\mathbf{x}_j) \parallel^2$
$= \kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_j \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{j,k} \alpha_k \alpha_j \kappa(\mathbf{x}_k, \mathbf{x}_j)$
$\Rightarrow \sum_i \alpha_i \parallel \Phi(\mathbf{x}_i) - c \parallel^2 =$
$\sum_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - 2\sum_{i,j} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sum_{j,k} \alpha_k \alpha_j \kappa(\mathbf{x}_k, \mathbf{x}_j) \Rightarrow$

$$\max_{\alpha} \sum_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{2.4}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{vl}, \sum_i \alpha_i = 1$$

All training examples with $\alpha_i > 0$ are support vectors, and the examples which also have a strict inequality for $\alpha_i < \frac{1}{vl}$ are considered non-bounded support vectors.

In order to classify a new test instance, $\mathbf{v}$, we would evaluate the following decision function:

$$f(v) = \kappa(\mathbf{v}, \mathbf{v}) - 2\sum_j \alpha_j \kappa(\mathbf{v}, \mathbf{x}_j) + \sum_{j,k} \alpha_k \alpha_j \kappa(\mathbf{x}_k, \mathbf{x}_j) - R^2$$

Before evaluating for a new point, R^2 must be found. This is done by finding a non-bounded support vector training example and setting the decision function equal to 0 [8]. If the decision function is negative for a new test instance, this

indicates a negative or healthy prediction. A positive evaluation is an unhealthy or positive prediction, and the magnitude of the decision function in either direction is an indication of the model's confidence.

2.2.3.3 An Experiment to Illustrate the Impact of Dimensionality on the One-Class SVM

In order to illustrate the impact of dimensionality on kernels and the one-class SVM specifically, an experiment with artificial data was constructed. Experiments with artificial data fill a critical gap by enabling modelers to entirely control datasets. Real-world data are required to validate techniques and demonstrate applicability. However, real world data often contains erroneously labeled instances, noise that may create spurious patterns, and human error. It is impossible to clean this data entirely. Testing performed with real data will always assumes these potential problems. Artificial data prevents these types of problems. If constructed properly, artificial data can truly test the limits of pattern recognition algorithms in a carefully controlled experimental environment.

In order to create data for the one-class SVM experiment, it was necessary to build a model of the pattern. This model is quite simple, involving standard normal distributions where the positive class and negative class have a difference of 2 between their means. This model can be presented as follows:

$$\mathbf{x}_{+1} = (z_1 + 1, z_2 + 1, z_3, ..., z_m),\ z_i \sim N(0,1) \text{ i.i.d}$$

$$\mathbf{x}_{-1} = (z_{1'} - 1, z_{2'} - 1, z_{3'} ..., z_{m'}),\ z_{i'} \sim N(0,1) \text{ i.i.d}$$

The true pattern only lied in the first two variables. All remaining variables were noise. Three types of kernels were examined. The linear kernel, polynomial kernel, and gaussian kernel.

Gaussian Kernel : $\kappa(\mathbf{x}_{+1}, \mathbf{x}_{-1}) = e^{-\|\mathbf{x}_{+1} - \mathbf{x}_{-1}\|^2 / 2\sigma^2}$

Linear Kernel : $\kappa(\mathbf{x}_{+1}, \mathbf{x}_{-1}) = \langle \mathbf{x}_{+1}, \mathbf{x}_{-1} \rangle$

Polynomial Kernel : $\kappa(\mathbf{x}_{+1}, \mathbf{x}_{-1}) = (\langle \mathbf{x}_{+1}, \mathbf{x}_{-1} \rangle + 1)^p$

Table 2.1: One Class SVM experiment for various dimensions on artificial data

Kernel	Parameter	dimensions	AUC	R^2
polynomial	p=1	2	0.874	12.8000
polynomial	p=1	5	0.8664	16.3600
polynomial	p=1	10	0.829	24.0600
polynomial	p=1	50	0.6642	73.2300
polynomial	p=1	100	0.618	127.9000
polynomial	p=1	250	0.5854	290.7000
polynomial	p=1	500	0.5683	551.1000
polynomial	p=1	1000	0.5549	1,064.0000
polynomial	p=2	2	0.7312	149.3000
polynomial	p=2	5	0.5946	359.8000
polynomial	p=2	10	0.5476	791.2000
polynomial	p=2	50	0.5014	5,918.0000
polynomial	p=2	100	0.4912	18,050.0000
polynomial	p=2	250	0.4952	88,640.0000
polynomial	p=2	500	0.5269	314,300.0000
polynomial	p=2	1000	0.5036	1,167,000.0000
polynomial	p=3	2	0.741	3,710.0000
polynomial	p=3	5	0.6065	10,490.0000
polynomial	p=3	10	0.5538	22,900.0000
polynomial	p=3	50	0.492	517,300.0000
polynomial	p=3	100	0.5172	2,634,000.0000
polynomial	p=3	250	0.5076	26,930,000.0000
polynomial	p=3	500	0.5111	179,700,000.0000
polynomial	p=3	1000	0.5117	1,291,000,000.0000
gaussian		2	0.9201	0.5149
gaussian		5	0.8978	0.4665
gaussian		10	0.8234	0.4356
gaussian		50	0.7154	0.3306
gaussian		100	0.6409	0.5234
gaussian		250	0.6189	0.4159
gaussian		500	0.5523	0.6466
gaussian		1000	0.5209	0.4059

The gaussian kernels in this experiment were tuned using an auto-tuning method. Typically for gaussian kernels, a validation set of positive and negative labeled data is available for tuning σ. In unsupervised learning, these examples of positive labeled data do not exist. Therefore, the best tuning possible is to achieve some variation in the values of the kernel without values concentrated on either ex-

treme. If σ is too large, all of the values will tend towards 1. If too small, they tend to 0. The auto tuning function ensures that the off-diagonal values for $\kappa(\mathbf{x}_{+1}, \mathbf{x}_{-1})$ average between 0.4 and 0.6, with a min value greater than 0.2 (notice that diagonal elements of a gaussian kernel matrix will always be one).

2.2.3.4 Multiple Classification Systems(MCS)

An interesting problem that exists with the one-class SVM involves the curse of dimensionality. This so called curse refers to the tendency for multivariate data to become equidistant as dimensions increase, preventing discrimination, classification, and clustering.

Given a classification problem of high dimension (greater than 10 variables), initial experimental results indicate that the creation of subspaces and aggregation of the subspace classification decision values results in improved classification over a model that utilizes all variables at once as shown in chapter 5 (the unsupervised classification model that will be utilized is the linear kernel one-class SVM from [24]). Creating subspaces for outlier detection, which is essentially what we are describing, is not a new concept [1]. However, considering this problem as a function of one-class SVM outputs and utilizing fuzzy logic for aggregation of output is a novel idea that this thesis explores.

A valid question would be to ask why not try a dimension reduction technique, such as principal components. Principal components work well with balanced classification problems, however caution is advised to use principal components as an overall dimension reduction technique for unsupervised problems. Often the difference between an intruder and non-intruder is subtle, and principle components along with other dimension reduction techniques often dilute the information content of the variables, especially in an unsupervised setting. Therefore, other techniques should be considered.

Bonissone et. al. explored the concept of applying fuzzy logic to the Multiple Classifier Systems (MCS) problem [15]. Their technique involved exploring the different aggregation techniques used in fuzzy logic, such as T-norms, to fuse decision variables. Chapter 5 details much of the recent work with multiple classification

systems. Multiple classification techniques presents an alternative to overcome this curse of dimensionality.

2.3 Receiver Operating Characteristic (ROC) Curves

A Receiver Operating Characteristics Curve is a simple and elegant way to display the performance of a binary classification system. ROC curves date back to World War II during the advent of radar. It was necessary to detect friendly planes from enemy planes, and the military needed reliable systems to perform this task. Continuing with our concept of referring to a true positive as the correct detection of something dangerous or malicious, it was desirable for the military to possess a system which performed at a high true positive (correctly identify an enemy plane) and low false positive (identify a friendly plane as enemy). The reasons for this is obvious. The medical community uses ROC curves extensively for measuring the accuracy of medical diagnosis tests [34–36].

Tom Fawcett published several papers regarding the application of ROC curves. He considers ROC curves and multiple classification systems in the context of game theory in [57]. In [55], he provides a very thorough theoretical review concerning in regard to everything that an ROC curve can (and sometimes cannot) represent. Kristin Bennett extended ROC theory for regression in the paper titled "Regression Error Characteristic Curves" [12].

ROC curves are a critical component of any binary classification problem. Classification systems create a real valued number as a decision value for each observed instance of the testing sample. Given a sample of test data, each data point will be assigned a decision value that typically ranges from -1 to +1. The data points that are true positive should fall closer to +1, where as the true negative cases closer to -1. The modeler must determine a threshold value for which to segregate and predict classes. The outcome in regard to true positive and false positive for one threshold value would correspond to one point on the ROC curve. If this threshold value is now incremented by some small value *epsilon*, from -1 to +1, a range of operating points will emerge. Plotting these operating points with respect to true positive (y axis) and false positive (x axis) will generate the ROC curve.

An ROC curve can also be considered a graphic representation of the relationship between the probability of a true positive outcome (sensitivity, 1-α error) and the probability of a false positive outcome (Type II error(β) or 1-specificity). The overall curve reflects the quality of the classification system. The area under the curve (AUC) is typically used as method of comparing alternate ROC curves; the better ROC curve typically has more AUC. Section 3.2 discusses in detail the meaning of the AUC, to include exactly how this statistic ties with the Wilcoxon Rank sum statistic and ranked data.

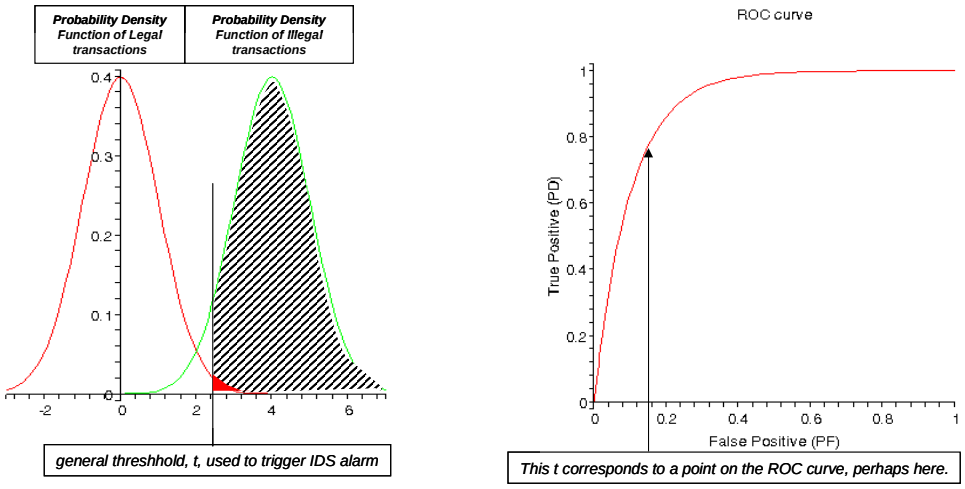

Figure 2.7: The plot on the left shows the PDFs of legal and illegal transactions (non intruders and intruders), respectively. The ROC curve on the right shows the possible operating point represented by the tolerance threshold shown on the PDF plots.

The plot of the probability density functions illustrates the idea of false positives and true positives. The hashed area represents the probability of a true positive, and the small red area represents a false positive. Imagine shifting the threshold, t, to the left. The true positive rating would definitely increase, but so would the false positive. The user must identify what point on the curve is acceptable, and this is usually accomplished through a cost-benefit analysis.

2.4 Recent Work in Computer Intrusion Detection

Computer intrusion detection systems involve either host-based intrusion detection or network-based intrusion detection. The first part of this section will cover recent work with host-based intrusion detection, detailing the Schonlau et. al., or SEA dataset. The second section will briefly cover recent work with network intrusion detection.

2.5 The Schonlau Dataset

2.5.1 Introduction to the Dataset

The SEA data provided a platform to explore host based intrusion detection. Before detailing exploratory effort with this data, the history and structure of the data will be explained. Matthius Schonlau collected this data from an AT&T lab in New Jersey, observing the UNIX computing behavior of 50 users. The data set consists of 50 users with each user contributing a stream of 15,000 truncated UNIX commands. The user's data stream is further divided into blocks of 100 commands, thus creating 150 blocks of commands for each user. The first 50 blocks of data is training data only (contains no masquerading data), and the remaining 100 blocks of data contains masquerading data that appears based on a probability. Given that there are 50 users, this implies that in total there are 2,500 tuples of data in the non-intruded initial set and 5,000 tuples of data that potentially contain intrusion data. Since the true outcome of each of these subsequent 5,000 tuples is known, understand that there are only 231 intruded tuples. Only 4.62% of the data contains simulated intruders, which makes this a very difficult problem. These simulated intruders involve streams of commands taken from other outside users (not part of the 50 authentic users). It is a very unbalanced classification problem. The objective is to determine if a block of data contains masquerading data or not.

2.5.2 Recent Work with the SEA Dataset

Schonlau et. al. [41–43, 128, 129] conducted the original work with this data. Their contributions included a thorough analysis of several statistical techniques for identifying masqueraders. Schonlau et. al. explored approaches that include: Bayes

one-step Markov model, hybrid multistep Markov model, text compression, Incremental Probabilistic Action Modeling (IPAM), sequence matching, and a uniqueness algorithm [41]. Schonlau stressed the importance of minimizing false positives, setting a goal of 1% or less for all of his classification techniques. Schonlau's uniqueness algorithm, explained in [129], achieved a 40% true positive rating before crossing the 1% false positive boundary. Wang [144] used one-class training based on data representative of only one user and demonstrated that it worked as well as multi-class training. Coull [30] applied bioinformatics matching algorithm for a semi-global alignment to this problem. Lee [101] built a data mining framework for constructing features and model for intrusion detection. Evangelista et. al. applied KPLS models as a supervised learning approach to the SEA dataset from several variables that they created for measuring user behavior [50].

Roy Maxion contributed insightful work with this data that challenged both the design of the data set and previous techniques used on this data [110, 111]. Maxion uses a 1v49 approach in [111], where he trains a Naive Bayes Classifier one user at a time using the training data from one user as true negative examples versus data from the forty-nine other users (hence 1v49) as true positive (masquerader) examples. Maxion claimed the best performance to date in [111], achieving a true positive rating of 60% while maintaining a false positive rating of 1% or less. Maxion also examines masquerade detection with a similar data set that contain command arguments in [110].

2.5.3 Network Intrusion Detection Systems (NIDs)

There are volumes of material that discuss network intrusion detection. It is a very broad field. NIDS is another vehicle to explore unbalanced binary classification. The NIDs problem involves collecting network traffic, which is typically collected as TCPdump packets [104], and these packets can be either stored for later analysis or inspected in an online fashion. Credit for some of the earliest work in intrusion detection goes to Denning who published a paper in 1986 that discussed a framework for monitoring system audit files [38]. Denning's original work served somewhat as a sentry call for the tremendous problem that computer security would become.

Since then, Denning wrote a book that covers many of the general threats that exist from the internet [37]. Numerous packet sniffing technology exists, and one of the more popular programs is Snort [2]. Snort is a rule based system that can employ both custom rule sets and generic rule sets that generally apply to all. Snort, and many of the other commercial NIDs, use signature detection. Known malicious code and malicious network attacks have a particular pattern, and once this pattern is understood it is simple to devise a technique to detect the pattern, or signature, and alarm appropriately. A more difficult problem involves detecting novel attacks, and significant progress is yet to be made towards this challenging problem.

As much as intrusion detection is actively pursued by academic researchers, it is probably more aggressively pursued by the commercial industry seeking practical implementations. Crothers wrote a book that takes a common sense, practical approach towards implementing network intrusion detection [32]. These types of works are becoming more common which is a clear indicator that the problem of intrusion detection is vast and organizations want solutions that can be implemented immediately. Ning, Jajodia, and Wang published some original concepts regarding distributed intrusion detection systems [82]. Distributed intrusion detection has increasingly become an important concept since attacks have become distributed (such as the distributed denial of service attack - ddos), and speed of detection has becoming extremely important. One of the commonly criticisms of intrusion detection is that intruders are not detected often until their deed is done, and that can be too late. Distributed detection may be a technique to improve upon this problem.

Some of the most popular research in NIDs involves the MIT Lincoln Lab DARPA testbed [68]. This testbed simulated network traffic that would be common to U.S. Air Force Base. Eight weeks of traffic was generated at 40 hours per week. This is almost a gigabyte of TCPdump data per week, which is significant if anything more than a trivial algorithm will be employed. Bernhard Sick analyzed this network traffic and generated 140 different variables, and from this set of variables he utilized evolutionary computing techniques to conduct variable selection [76]. Most of the testing conducted with this data revealed the obvious -

intrusion detection systems were very good at identifying previously known attacks, and very poor with anything novel. Marchette wrote a text that explores network monitoring and intrusion detection from a statistical viewpoint [108]. His work discusses the fundamentals of network data, techniques to quantify the data, and methods for detecting malicious activity.

CHAPTER 3
THE RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

Given the importance of the ROC curves to intrusion detection and binary classification problems as a whole, it is worthwhile to dedicate a chapter to discuss the aspects of this curve. ROC curves are widely studied and utilized for the evaluation of machine learning applications and medical classification and decision making [55]. Much of the appreciation for ROC curves stems from the complete display binary classifier performance across the full spectrum of thresholds. This is particularly important for unbalanced problems where performance in particular ranges of the ROC curve are of interest. The medical community studies ROC curves largely to better understand the performance of medical tests and the statistics which can support a diagnosis [134]. This chapter concludes by introducing the *decision ROC chart*, a novel method for illustrating the relationship between ROC curves and decision values.

Before further discussion, let us define some notation. $\mathbf{x}_i$ will represent a m-dimensional pattern or observation, and y_i will represent the associated label or class of this pattern (for binary classification $y_i \in (-1, 1)$). $\hat{y}_i \in \mathbb{R}^1$ will represent the decision value created by the machine learning algorithm, and $R_i \in (1, 2, ..., N)$ will represent the rank of the decision value, an important number when constructing ROC curves.

3.1 Confusion Matrices

ROC curves are a two dimensional graph, plotting the true positive (TP) rate on the y or vertical axis and the false positive (FP) rate on the x or horizontal axis. The confusion matrix is closely related and essentially a subset of an ROC curve. The confusion matrix displays four numbers - true positive(TP), true negative(TN), false positive(FP), false negative(FN) - which illustrate the prediction performance of a classifier at a specific threshold. Table 3.1 shows a typical confusion matrix.

		Predicted Label	
		positive	negative
Actual	positive	TP	FN
Label	negative	FP	TN

Table 3.1: The Confusion Matrix

From this confusion matrix there are a number of statistical performance metrics which can be derived. All of these metrics are based upon the binary classification problem. These metrics are also defined in [92].

$$\begin{aligned}
&\text{Accuracy} = && \frac{TP+TN}{TP+TN+FP+FN} \\
&\text{True Positive Rate(Recall, Sensitivity)} = && \frac{TP}{TP+FN} \\
&\text{True Negative Rate(Specificity)} = && \frac{TN}{TN+FP} \\
&\text{Precision} = && \frac{TP}{FP+TP} \\
&\text{False Positive Rate} = && \frac{FP}{FP+TN} \\
&\text{False Negative Rate} = && \frac{FN}{FN+TP}
\end{aligned}$$

The medical community uses the terms specificity and sensitivity, and it is evident that sensitivity is equivalent to the true positive rate and specificity is equivalent to (1 - false positive rate). It is quite common for medical researchers to illustrate medical test performance through plotting sensitivity versus specificity. The more common ROC curve plots the false positive rate on the horizontal axis and the true positive rate on the negative axis.

A confusion matrix can also display the accuracy of a multi-class problem, often illustrating model tendencies and which classes the model tends to "confuse". The simplest way to better understand confusion matrices is through an example.

Figure 3.1 is a toy dataset that will be used to illustrate several concepts in this chapter. Let us assume that a threshold of 0 is applied to this dataset, creating the confusion matrix shown on the right side of figure 3.1.

Decision Value ($\hat{y}_i$)	**Rank** (R_i)	**True Class** (y_i)
2.893	1	1
2.208	2	1
1.664	3	1
0.991	4	1
0.889	5	-1
0.609	6	1
0.015	7	1
0.013	8	-1
-0.240	9	-1
-0.278	10	1
-0.808	11	-1
-1.257	12	-1
-1.437	13	-1
-1.750	14	-1
-1.864	15	-1

		Predicted Label	
		positive	negative
Actual Label	positive	6	1
	negative	2	6

Figure 3.1: Toy dataset with sorted decision values, ranks, and true classes. Confusion matrix illustrates performance with a $\hat{y}_i$ threshold of 0.

3.2 Algorithms for Creating an ROC Curve and Finding the Area Under the Curve

The simplest and most conceptually straightforward algorithm to create an ROC curve is to build confusion matrices that illustrate performance for a range of threshold values. These confusion matrices will each represent a point on the ROC curve; each confusion matrix will have a unique point on the curve created from the false positive rate and true positive rate that the confusion matrix represents. In order to create an empirical ROC curve with the best granularity, a confusion matrix could be created at a threshold boundary which separates two distinct observations. It is possible to define these thresholds through either the ranks or actual decision values. As will be shown in later algorithms, it is more more elegant to manage the ranks. Algorithm 4 illustrates how to create ROC curves based on confusion matrices.

Fawcett details algorithm 5 in [55], a method which creates an ROC curve based solely on ranks . Algorithm 4 is a much more intuitive method, however algorithm 5 is much more elegant and compact, and requires less computation.

Algorithm 4 Creating an ROC curve from N confusion matrices.

Reverse rank order all observations by $\hat{y}_i$, assigning the smallest rank (1) to the largest $\hat{y}_i$, $i \in (1, 2, ..., N)$
Let p represent the total number of positive instances and b represent the total number of negative instances
for $R_i = 1$ to N **do**
  $t = R_i$, $TP = 0$, $FP = 0$
  for $R_j = 1$ to t **do**
    if $y_j = 1$ **then**
      $TP = TP + 1$
    else if $y_j = -1$ **then**
      $FP = FP + 1$
    end if
    $k = R_i$
    True Positive Rate $(TPR_k) = TP/p$
    False Positive Rate $(FPR_k) = FP/b$
    Store or plot (FPR_k, TPR_k) as k^{th} point on ROC curve
  end for
end for
Let $TPR_0, FPR_0 = (0, 0)$

Algorithm 5 Creating an ROC curve solely based on ranks.

$FP = 0$, $TP = 0$
Let $(FPR_0, TPR_0) = (0, 0)$
for $R_i = 1$ to N **do**
  if $y_i = 1$ **then**
    $TP = TP + 1$
  else if $y_i = -1$ **then**
    $FP = FP + 1$
  end if
  $k = R_i$
  $FPR_k = FP/b$
  $TPR_k = TP/p$
  Store or plot FPR_k, TPR_k as k^{th} point on ROC curve
end for

ROC curves of good classifiers typically have a large convex rise which approaches the (1,1) point. It is also possible that ROC curves will indicate poor classifier performance at false positive or true positive ranges. In the security classification problem, our primary interest involves improving performance in the low false positive range. An ROC curve which simply plots along the diagonal, creating an AUC of approximately 0.5, indicates a classifier which contains no predictive power; this ROC curve indicates random classification, no better than tossing a coin. It is also possible to have a concave ROC curve, which usually indicates a reverse classification or reverse labeling problem.

Rank (R_i)	**True Class** (y_i)
1	1
2	1
3	1
4	1
5	-1
6	1
7	1
8	-1
9	-1
10	1
11	-1
12	-1
13	-1
14	-1
15	-1

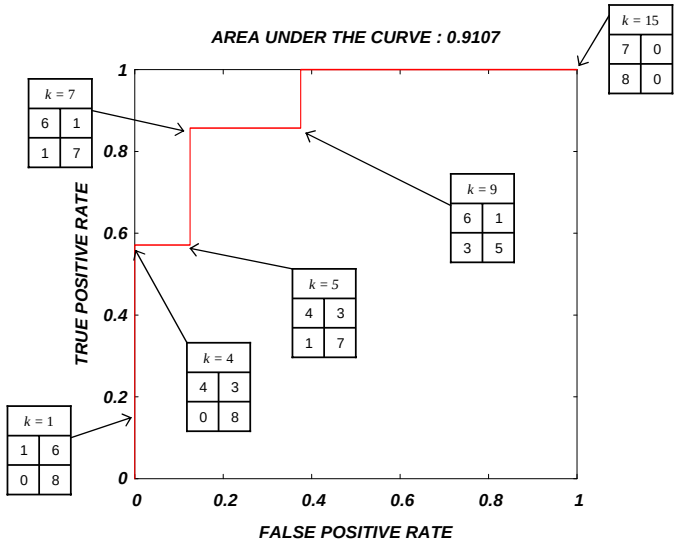

Figure 3.2: An example of the ROC curve created from the toy dataset as per algorithm 4. The values in the confusion matrices are consistent with table 3.1.

It is also of interest to find the area under the ROC curve (AUC), as this is a good overall performance metric for a binary classification problem. The most obvious way to find this area is through numerical integration of the curve found in algorithm 4. Algorithm 6 uses a sum of trapezoidal areas to estimate the AUC.

It is also well known that the AUC has a special probabilistic meaning. The AUC is equivalent to the probability that a randomly selected positive instance ranks above (has a smaller R_i) than a randomly selected negative instance. Let

Algorithm 6 Finding the Area Under the ROC Curve (AUC) from Numerical Integration.

1: $AUC = 0$
2: **for** $k = 1$ to N **do**
3: $\quad A = .5(TPR_k + TPR_{k-1})(FPR_k - FPR_{k-1})$
4: $\quad AUC = AUC + A$
5: **end for**

R_i, $i \in (1, 2, ..., p)$ represent the rank of positive instances, and R_j, $j \in (1, 2, ..., b)$ represent the rank of negative instances. This probability is also known as the Mann-Whitney U statistic, a statistic we will use to find the AUC.

$$P(R_i < R_j) = \text{AUC} = U$$

Much of the original rank statistics studies which apply to ROC curves can be attributed to Wilcoxon, Mann, and Whitney who used ranks to measure whether or not two random variables were statistically different. Wilcoxon's work was a cursory exploration to "obtain a rapid approximate idea of the significance of the differences in experiments" [145]. The dataset utilized in his experiments measured the lethality of two different fly sprays. Mann and Whitney also used rank statistics to determine the statistical difference between two treatments, with much more of an emphasis on describing the underlying distribution of the U statistic [107]. Lehmann provides a comprehensive study of rank statistics in his book *Nonparametrics: Statistical Methods Based on Ranks*, thoroughly discussing the Wilcoxon Rank Sum statistic and Mann-Whitney U statistic [102].

Let $R_i, i \in (1..p)$ represent the rankings of actual attacks, and let $R_j, j \in (1..b)$ represent the rankings of the workstations which were not actually attacked. The Wilcoxon Rank Sum Statistic, W_s, is equivalent to $\sum_{i=1}^{p} R_i$. Since the sum of all rankings is $\frac{1}{2}N(N+1)$, it follows that the sum of non-attack instance rankings is $W_r = \frac{1}{2}N(N+1) - W_s$. W_r can also be calculated as shown in equation 3.1 [102].

$$W_r = \sum_{j=1}^{b} R_j = \frac{1}{2}b(b+1) + \sum_{i=1}^{p}\sum_{j=1}^{b} \varphi(S_i, S_j) \tag{3.1}$$

$$\text{where } \varphi(R_i, R_j) = \begin{cases} 1 & \text{if } R_i < R_j \\ 0 & \text{otherwise} \end{cases}$$

The statistics $W_{XY} = W_s - \frac{1}{2}p(p+1)$ and $W_{YX} = W_r - \frac{1}{2}b(b+1)$ are also popular forms of the Wilcoxon Rank Sum statistic, and it is this form of the statistic that relates to the area under the ROC curve. The Mann-Whitney U statistic, which is exactly equal to the area under the receiver operating characteristic curve, is directly proportional to the Wilcoxon rank sum statistic W_{YX} and shown in equation 3.2.

$$U = \frac{W_{YX}}{pb} = \frac{1}{pb}\sum_{i=1}^{n}\sum_{j=1}^{p}\varphi(R_i, R_j) = \text{AUC} \tag{3.2}$$

Hanley and McNeil show in [71] that the area under the ROC curve equates to the Mann-Whitney U statistic. Realizing this, algorithm 7 can be used to find the AUC.

Algorithm 7 Computing the Mann-Whitney U statistic.

```
W_r = 0
for i = 1 to N do
    if y_i = −1 then
        W_r = W_r + R_i
    end if
end for
W_YX = W_r − (1/2)b(b + 1)
AUC = U = W_YX / pb
```

ROC curves are not the panacea for measuring the performance of binary classification. Although these curves convey significant information, it is important that researchers understand the limitations and properties of these curves. Many authors, to include Fawcett [55], Bradley [17], and Mason and Graham [109], advise due caution when using ROC curves to measure the performance of binary classifiers. ROC curves are non-parametric. There is value in the simplicity and pragmatism of displaying a classifier's output as non-parametric ranks, however it is without doubt that information is lost when we reduce the decision value from the classifier to a rank.

3.3 Sub-ROC curves

Quite often researchers may only have concern with a portion of the ROC curve. In the medical community, false negatives are of major concern. It is not acceptable to have a test commonly dismiss a diseased patient as healthy. Therefore, physicians are likely to prefer operating in a relatively high false positive range; false positives are tolerable, but false negatives are not. In the security domain, false positives can become a significant problem. This is due to the extreme imbalance between the classes in this problem. In the computer network security domain, there are significantly more benign connections compared to the number of malicious connections. A large percentage of false positives can create an overwhelming load of benign connections tagged as malicious, each of which must be examined. If the false positive load becomes too burdensome, the system no longer creates value and simply turns into a bottleneck which end users quickly abandon.

Therefore, it is quite useful to study a portion of the ROC curve, especially if it is likely that this is the only region which an application will ever operate, the rest of the ROC curve is relatively meaningless. McClish was likely the first researcher to write about the concept of partial ROC curves. McClish's method assumes a binormal distribution between the two classes which the classifier is attempting to separate, and her interest involves comparing two ROC curves only within a certain false positive range [112]. Jiang et. al. analyzed partial ROC curves for specific ranges in sensitivity, or true positive [84]. Zhang applied the same non-parametric method established in the Mann-Whitney U statistic to only a portion of the ROC curve [152]. This non-parametric approach to partial ROC curves is the method which will be discussed here.

The following discussion of sub-ROC curves and the partial AUC utilizes statistics derived during the course of this research. These statistics are not claimed as novelties of this thesis. The partial AUC is the truly creative component of this theory and this concept has been studied by others. ROC stickiness, although a relatively simple concept, has not been mentioned in the literature reviewed and could be claimed as a novelty of this work.

One of the simplest ROC metrics involves calculating what type of true positive

rate to expect before observing a single false positive. Counting from the top of the rankings until reaching the first negative instance, let R_i^* be the rank of the last positive instance. The *stickiness* of the ROC curve (*stickiness* refers to the ROC curve sticking to the 0 FP line) can be defined as:

$$\text{ROC } \textit{stickiness} = \frac{R_i^*}{p}$$

The next step beyond sticky ROC curves involves the partial AUC, when a portion of the false positive range is of interest. In the security classification problem, the typical ROC curve region of interest is the low false positive domain. Some authors insist that good performance must occur at a very small false positive range, such as Schonlau did in [41].

Suppose that in a given dataset, b negative instances exist and p positive instances exist. If the false positive tolerance, or false positive point of interest (FPPOI) is a threshold which is not acceptable to cross, then we can define:

$$\text{FPPOI} = \frac{b'}{b}$$

b' now represents the number of false positives for a specific dataset which define the threshold. It is also possible now to define the following additional statistics:

$$W_r' = \sum_{j=1}^{b'}$$

$$W_r' - \frac{1}{2}b'(b'+1) = W_{YX}'$$

$$(W_{YX}'/p'b')(p'/p)(b'/b) = \frac{W_{YX}'}{pb} = \text{ AUC up to FPPOI}$$

The AUC up to the FPPOI is the partial AUC, the area under the ROC curve up to the FPPOI. It also may be of interest to observe what the Mann Whitney U statistic is for that region of the ROC curve. For example, given this small fraction of observations up to the FPPOI, what is the probability that a positive instance is

ranked higher than a negative instance? This can be calculated by simply managing this fraction of observations as its own set of rankings and its own ROC curve. It has already been established that there are b' negative instances and p' positive instances in this subset. Let $R'_j, j \in (1, 2, ..., b')$ define the ranks of the negative instances in the subset, and let $R'_i, i \in (1, 2, ..., p')$ define the ranks of the positive instances in the subset.

$$P(R'_i < R'_j) = \frac{W'_{YX}}{p'b'}$$

Rank (R_i)	**True Class** (y_i)
1	1
2	1
3	1
4	1
5	-1
6	1
7	1
8	-1
9	-1
10	1
11	-1
12	-1
13	-1
14	-1
15	-1

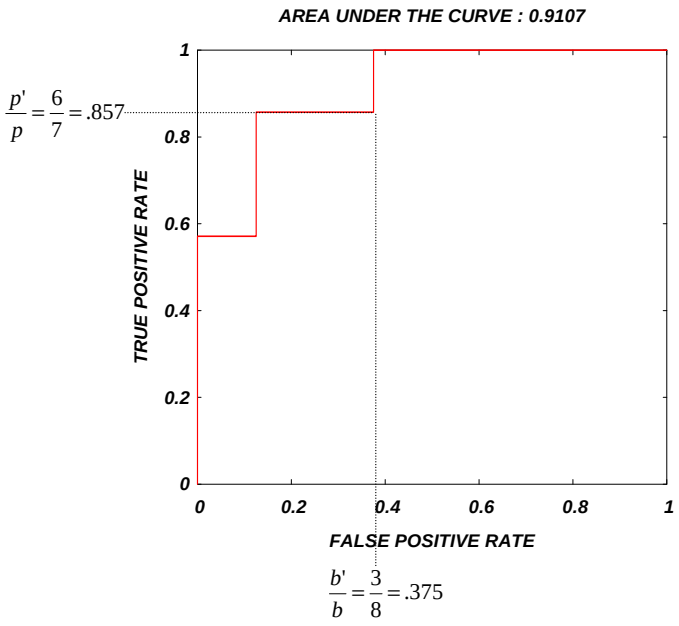

Figure 3.3: Illustration of the partial AUC

An example with the toy dataset shown in figure 3.3 will further illustrate the sub-ROC curve concept. Assume that our false positive tolerance is .375, implying a FPPOI = $b' = 3$ and $p' = 6$. From previous equations, $W'_r = 5 + 8 + 9 = 22 \implies W'_{YX} = 16$. The AUC up to the FPPOI is $16/(8\times9) = .286$. It also follows that the AUC of a sub-ROC curve, considering only these 9 instances as the total number of observations, would equate to $16/(3\times6) = .89$.

The probabilistic meaning of the area under the ROC curve enables researchers to create ROC curves that possess the exact types of attributes of interest, without the need for any data or prediction modeling. These ROC curves, created strictly

from pseudo-random numbers, are a novelty of this research and included in section 6.3. The analysis of these pseudo ROC curves, and their underlying rank distributions, provided a solid foundation to show why certain types of fusion methods created synergistic results. Further discussion of this can be found in chapter 5.

3.4 Pseudo ROC curves

Many of the fundamental concepts regarding ROC curves from this chapter are applied in chapter 5 to introduce one of the significant novelties of this research. Chapter 5 discusses the probabilistic meaning of the AUC and utilizes this property to create pseudo-ROC curves and rank distributions. Creation of pseudo-ROC curves and rank distributions provide the underlying theory for ensemble methods designed specifically for the security classification problem. The detailed discussion of pseudo ROC curves is left to this future chapter.

3.5 Improved Decision Making with the *Decision ROC Chart*

The eventual purpose of an ROC curve is to support a decision, however ROC curves provide an incomplete representation of the decision environment. Important information not included in ROC curves include balance of classes and decision values. Practitioners value ROC curves for the insight that these curves provide to a classifier or detection device, however applying information from the ROC curve is difficult without some type of translation from ROC space to the decision space. The decision space for a binary classification problem involves considering the output of a model, $\hat{y}_i$, and comparing this output with a threshold, t.

Classifiers create decision values. These decision values, $\hat{y}_i$, exist on a spectrum in the real number realm ($\hat{y}_i \in \mathbb{R}^1$), where $\hat{y}_i$ represents nothing more than the classifier's judgment on the class membership of an observation. Classifiers base this judgment on some type of function created from other observations, so the decision value becomes the classifier's similarity or strength of belief metric indicating class membership. Decision values often contain no true meaning. Many decision values follow some type of sigmoidal function, clustering in the center and existing in a lower density at the extreme ends of the the spectrum. For some types of detec-

tion equipment, decision values could represent the output of some type of sensor. This could be a chemical measurement as in a medical test or an electronic signal measurement which exists in many types of detection scenarios. Regardless of the classification problem, the end result is that a decision must be made whether to classify an unlabeled instance as positive or negative. ROC curves illustrate the performance of a classifier, but they do not provide a complete picture or aid in classification decisions. Practitioners must bridge the gap between ROC curves and decision values. The following extension to the ROC curve is a simple and novel method to aid decision makers and assist in bridging the gap from ROC curves to decision values.

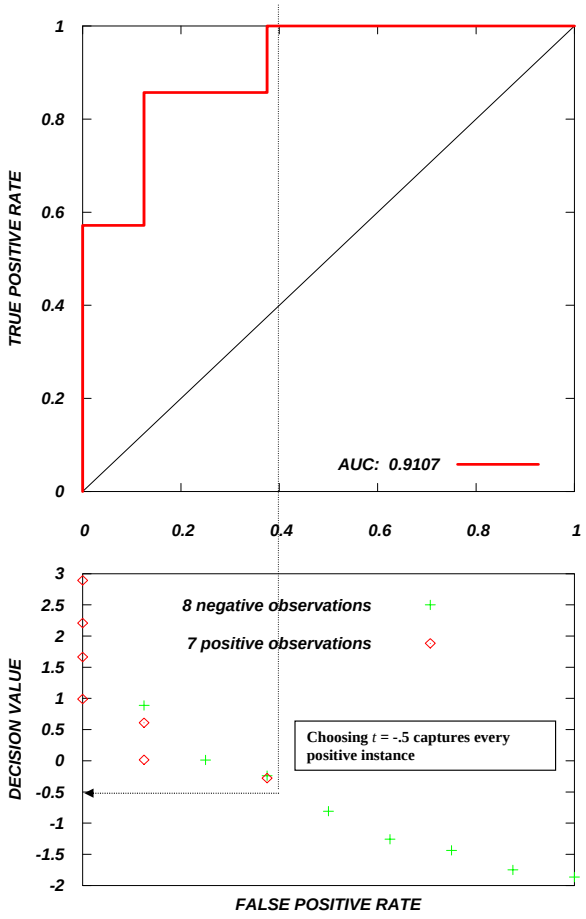

Figure 3.4: *Decision ROC Chart* from data in figure 3.1

The horizontal axis of an ROC curve measures the false positive rate of a classifier, and this false positive rate is simply a fraction of negative instances.

Referring to table 3.1, recall that the False Positive Rate = $\frac{FP}{FP+TN}$. The extension to the ROC curve, referred to as the *Decision ROC Chart*, plots the false positive rate on the horizontal axis, enabling the bridge between the ROC curve and the decision value. The vertical axis represents the decision value, $\hat{y}_i$. The *Decision ROC Chart* captures much of the information absent from the ROC curve, to include the balance of the problem, distribution of the decision values, and perhaps most importantly the connection between the the ROC curve and the decision value through the false positive rate.

Figure 3.4 illustrates the *Decision ROC Chart* for the toy running problem discussed in this chapter from starting in figure 3.1.

Figure 3.5 illustrates the *Decision ROC Chart* for a larger dataset. In addition to aiding the threshold decision for a classification problem, the *Decision ROC Chart* provide clarity to ROC curves for those who are not familiar with ROC curves. The area under the curve is visibly seen as a probability with the *Decision ROC Chart*. It is much more believable, and understandable, that the probability of a positive value out ranking a negative value is 0.959.

Decision ROC Charts educate decision makers and students involved with classification. The purpose of these curves is simply to educate, assist in decision making, and provide a more complete picture of a decision environment.

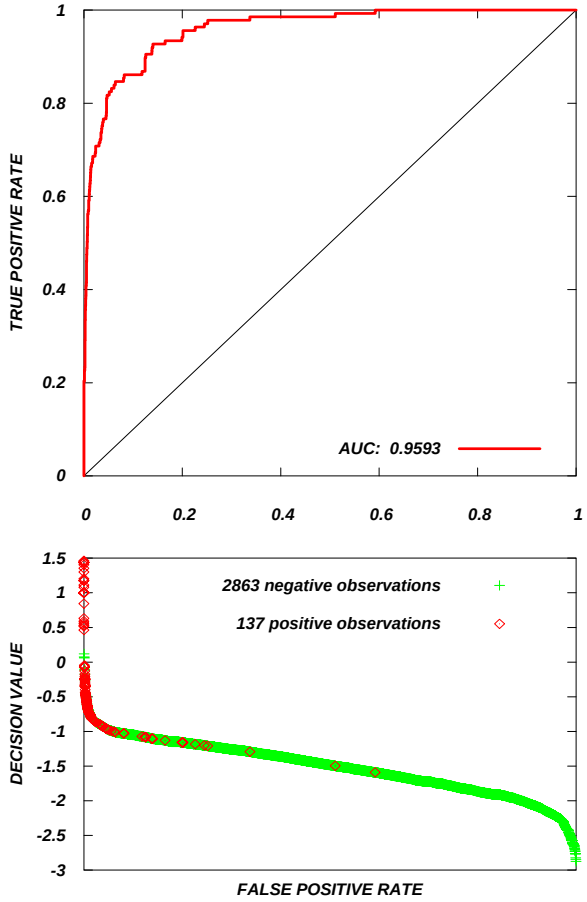

Figure 3.5: Decision ROC Chart for a large dataset

CHAPTER 4
COMPUTER INTRUSION DETECTION VARIABLES

This chapter discusses how to harvest variables for machine learning from host-based monitoring and network-based monitoring. The methods introduced are techniques that worked. Other methods, perhaps even better techniques for creating variables, could be possible. This is a chapter designed to illustrate how to transform measurements from a security environment into data that models the environment. This chapter introduces the necessary first preprocessing steps that must be accomplished before applying techniques described in subsequent chapters of this dissertation.

4.1 Information and Computer Security

There are numerous techniques used to enhance the security of information and computers. Many of these techniques attempt to block or deter attackers, preventing them from intruding in the first place. These techniques include many of our everyday encounters such as user-accounts and passwords, firewalls, and VPN encryption. All of these techniques aim to keep the wrong people out and the right people in. However, attackers continually strive to break through these barriers and invade protected information. These barriers are preventive measures, and digital firms must augment these preventive measures by implementing intrusion detection systems (IDS) that monitor the network and detect fraudulent or unusual activity.

Certain characteristics of computer IDS define their effectiveness. The typical measures of an IDS involve accuracy of detection, often expressed as a true positive rating and false positive rating. The overall effectiveness of an IDS can be shown on a receiver operating characteristic (ROC) curve, which plots the full range of a classification system's true positive and false positive rates. Typically, minimizing false positive alarms while maintaining an acceptable rate of true positives is paramount with intrusion detection systems. Falsely identifying a significant portion of authentic users as intruders represents a problem for a digital firm. Authentic users could experience temporary restriction or no access to the network. This creates

frustration on an individual level and could result in reduced efficiency on a larger scale. Minimizing false positive ratings is a key criterion when evaluating an IDS, even while realizing that a lower true positive rating will occur.

Intrusion detection is inherently a statistical problem [41]. The problem involves collecting a sample of data, describing the data through statistical attributes, and then classifying the data based upon these attributes. Intrusion detection oftentimes involves identifying intruders without any training data that indicates the behavior of intruders.

4.2 Types of Intrusion Detection Systems

There are two types of intrusion detection systems: host-based IDS (HIDS), and network-based IDS(NIDS). Host-based IDS monitor individual workstations, and Network-based IDS monitor network packets. The types of intrusion detection used for these types of IDS are inherently different due to the type of data generated by each. Computer workstations monitored by a HIDS can generate very specific data regarding user behavior. Networks monitored by HIDS produce one type of data: network packets.

Monitoring with HIDS is a domain specific task. The purpose of using a HIDS is to prevent an individual from masquerading as an authentic user and to prevent authentic users from wandering into computing domains where they do not belong. Smaller banks of computers with consistent users operating in a relatively secure environment would be an appropriate domain for a HIDS. The intent of the HIDS is to capture telling statistics about user behavior that helps to identify a particular user. This information is then transformed into some type of statistic or set of statistics that can be used as a benchmark to compare the future behavior of the same user. If this user's behavior strays far enough from this statistic, the HIDS will alarm an intruder. The idea is that computer users have certain behavior, and often patterns of behavior, that they perform every time they log onto a machine. Significant deviation from this behavior indicates a potential intruder. The types of data that can be collected by a HIDS included keystroke statistics, program preferences, websites visited, times logged on, command usage, etc. HIDS intrusion detection

often is considered anomaly detection since the purpose is to detect anomalies or outliers measured against historical statistics.

Monitoring a network with a NIDS is somewhat different. The purpose of a NIDS is to identify network traffic indicative of network attacks. Examples of this include individuals attempting to attack into a company's intranet and computers inside of the network attempting to attack from within. NIDS typically utilize some type of packet sniffing software, such as Snort [2], which has the capability to filter packets based upon certain decision criteria. A NIDS will typically pass any packets deemed as benign and analyze all of the packets identified as a potential attack. NIDS utilize known characteristics established from known attacks, and for this reason these techniques are often referred to as signature based detections. Previous attacks typically have a certain signature, and a NIDS attempts to identify matching or similar signatures.

4.3 Analysis of the Host Based Dataset

Section 2.5.1 provides a detailed account of the contents of the SEA dataset. Although widely used for masquerade detection, this dataset has some limitations [41]. First, command arguments, which may contain valuable information for intrusion detection and authorship, were not collected because of privacy concerns. Second, the 20 intruders are "other" users from a general population similar to the legitimate users and not "real" masqueraders who may exhibit unusual patterns distinguished more easily from normal patterns. Third, mistyped commands may violate the signature profile built for a user. Regardless of these shortcomings, we found the data to be a valuable set and used it to test our methods.

The initial programming effort involved determining the data dictionary (list of every unique command). In addition to generating this list, the programs also report the frequency of the commands and the popularity of the commands (number of users who use distinct command). In an effort to validate the data dictionary, frequency, and popularity of the commands, we relied upon a graph created by Schonlau that plots distinct commands vs. unique commands [129].

Figure B.4 contains Schonlau's plot from [129] on the right and an identical plot

used to validate analysis of the data on the left. These plots illustrate the uniqueness (1 - (total users of that command / total users)) of the distinct commands. As shown, almost 50% of the distinct commands have a uniqueness of .98, meaning that almost 50% of the distinct commands are used by only one user. In total, there are 856 distinct commands within the entire data set, and within the training data set (first 5000 commands for each user) there are 635 distinct commands.

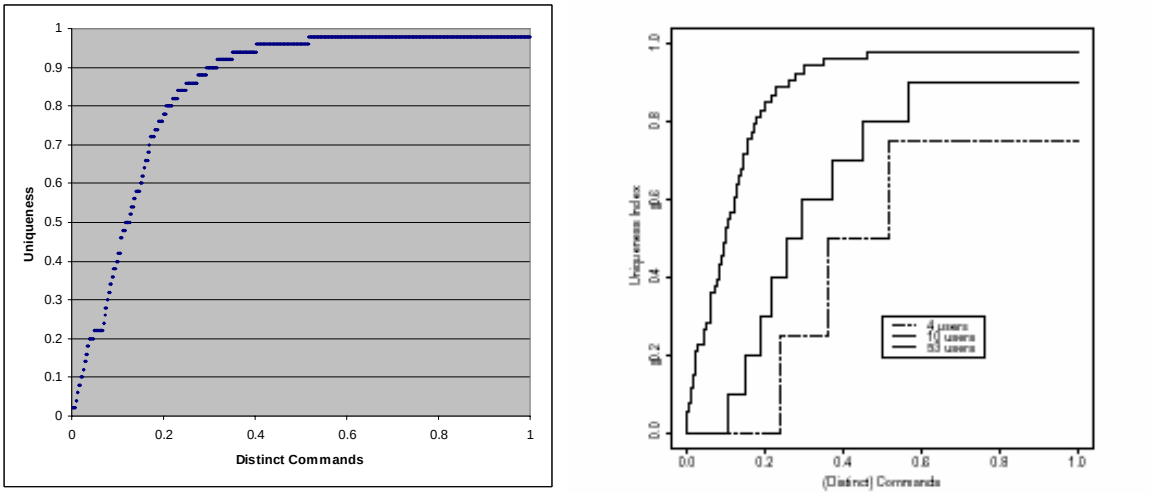

Figure 4.1: Distinct commands plotted vs. unique commands

4.4 Schonlau's Analysis

Before creating text mining variables, much of Schonlau's original work and results with this data were recreated. Schonlau developed, analyzed, and compared numerous detection algorithms. Schonlau's work includes analyzing unpopular commands, real time hypothesis testing, principal components regression, and structural zeroes [41–43, 128, 129].

Following the work of Schonlau accomplished two critical goals. Replicating his work and achieving the same results through new text mining programs written in Perl completed a validity check for the programs. Secondly, his work with this dataset is comprehensive, and understanding and replicating his work provides a solid foundation of knowledge that enables further exploration of this dataset.

Shortly after finishing the initial programming and validity check with Schon-

lau's algorithm based upon unpopular commands [129], new techniques were explored. It is possible to measure this data in numerous ways and provide variables that a learning machine can use to predict outcomes. After creating a new set of variables that measure user behavior, these variables serve as a basis for experimentation in predictive modeling with learning machines. The primary learning machine utilized in this article is Rosipal's K-PLS [124]. Although other learning machines may offer comparable or even better results, our choice to remain with this particular learning machine is primarily an effort to remain consistent. With numerous combinations of variables presented to the learning machine with different preprocessing techniques, exploration of improved classification begins.

4.5 Text Mining Variables

After validating the Perl programs and developing a solid understanding of the dataset, new variables were necessary to create different techniques that accurately predict the presence or absence of an intruder. The spirit of these variables come from Schonlau's original work [129], however the machine learning approach involving these variables is fundamentally different from Schonlau's technique. These variables were created with the goal of developing a technique to measure the behavior of computer users in a manner that captures the differences between intruders and authentic users. There are a number of techniques available to capture computer user behavior and/or network behavior. The selected technique is largely a function of the dataset utilized. The new variables and the prediction models involving these new variables work primarily with the latter 10,000 commands from each user. Therefore, with each user contributing 100 tuples of data for this section, there are a total of 5,000 tuples that we will examine. These new variables we are discussing measure the overall characteristics of the 100 commands contained within the tuple, and these variables can be seen in table 4.5.

An important aspect that remained true throughout the search for effective classification involved the importance of Mahalanobis scaling the data - essentially normalizing every entry in the data matrix - by subtracting the mean and dividing by the standard deviation of the variable. This was due to the extreme ranges

Table 4.1: Description of text mining variables

	Name of Variable	Description
1	new	Binary variable that indicates presence of command never used by this user
2	% of Unix commands	Percent of commands that are UNIX
3	% Top 20 most popular commands	Fraction of commands that are within the top 20 most popular commands in the data set
4	% of internet commands	Fraction of commands that are internet/email commands (sendmail and netscape)
5	Average Uniqueness	Averages Uniqueness (number between .02 and .98) of commands in tuple
6	Average Frequency	Averages the frequency of each command in the dataset
7	% of foreign commands	Percent of commands never seen before - this will be zero for all training data
8	x_u value	x_u value from Schonlau's algorithm[8], essentially providing a signature index that is based upon usage pattern of that particular user
9	user	Integer between 1 and 50 that indicates the identity of the user

of different variables; with similar reasoning, the covariance matrix provides little useful information, however the correlation matrix provides a true measure of the interdependency between variables.

4.6 Matrix Plots for Visualizing Variables

It is sometimes possible to predict a response directly from the values of the variables, without any learning machine or algorithm. If a variable takes on a certain range of values for one response, and a separate range for another response, it is sometimes possible to infer with a good degree of certainty the response based on a single variable or multiple variables that have this property. "Fisher's Iris data", shown in Figure 4.2, is a great illustration of this point. The iris dataset is a classic multivariate dataset originally analyzed by Fisher in [58] and also included in the work of Johnson and Wichern [85]. Four predictor variables: petal length, petal width, sepal length, and sepal width are used to distinguish types of iris flowers. The $Analyze^{TM}$ [46] software package contains an option to color map observations from the dataset. This is very similar to the color map PM3D process available

from gnuplot [140].

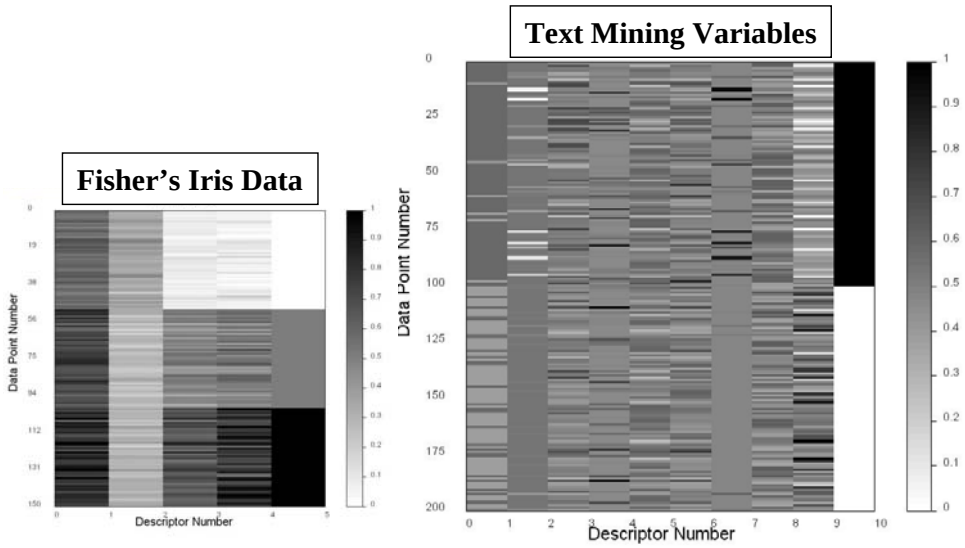

Figure 4.2: Matrix plots for the description of variables

In the Iris dataset it is apparent that the sepal length and width (first two variables) will not be as effective determine the type of iris since the colors do not have a distinct similarity with those in the response column. On the other hand, the colors generated for the petal length and width are distinctly similar to the response column. A reasonable assumption is that petal length or petal width can be used to fairly accurately predict the type of iris. The computer intrusion dataset was sorted by the "Intrusion/ Non-Intrusion" column and color coded in the same way as the Iris data set. These new variables introduced for the computer intrusion detection problem do not exhibit this type of behavior, except for Schonlau's x_u value from the uniqueness algorithm(7th variable shown adjacent to output variable) and the 1st variable, "new", which is a binary variable which represents the presence or absence of commands never seen from a particular user. This is expected because the x_u value is an overall index that predicts the presence or absence of an intruder; low x_u values correspond with potential intruders, and higher x_u values indicate authentic users, hence the inverse relationship illustrated. The correlation of the "new" variable and the output variable is a direct reflection of the consistent and stagnant command vocabulary of most of these users.

4.7 Analysis of the Network-Based Dataset

The analysis of network-based data is an entirely different problem from hos-based intrusion detection. The network-based data are much more voluminous, and also contain many more dimensions. It is important to initially have a basic understanding of network traffic. Network traffic consists of packets of information sent from a host to a destination. The most popular protocol today is transmission control protocol / internet protocol (TCP/IP). This protocol breaks messages into packets, labels the packets, and routes the packets to a host. The receiving machine ensures all packets are received and assembled correctly based upon this protocol. TCP/IP is what makes the internet work. There are other protocols, such as user datagram protocol(UDP) and address resolution protocol(ARP), which exist on the network, however they are not nearly as prevalent or important as TCP/IP.

The typical collection format of network data is TCPdump. TCPdump is a UNIX program that collects packets from a network line. This is usually a router or gateway when attempting to monitor a network. Collecting and analyzing TCP-dump data is the fundamental method of analyzing network traffic.

The primary dataset for examining network intrusion detection is the 1998 and 1999 DARPA Intrusion Detection Evaluation [68]. This evaluation took place at the Lincoln Laboratory at the Massachusetts Institute of Technology. The MIT researchers collected several weeks of data from an Air Force base that was not exposed to any known network attacks. The researchers then artificially inserted attacks within this data through simulation. The end result was a dataset with known attacks of several types, and the start and end of these attacks is known and marked. This is exactly the type of information necessary for machine learning research. However, there is a significant challenge involving the representation of this data.

TCPdump data is alphanumeric data presented as a header followed by the payload of data. An example of five different TCPdump packets are shown in figure 4.3.

There are several problems with directly analyzing TCPdump packets. The first problem is the volume. An average eight hour day of traffic from the DARPA

```
08:04:27.669417 197.182.91.233.smtp > 172.16.112.207.1027: P 1:84(83) ack 1 win 32736 (DF)
                4500 007b 040f 4000 4006 f7ee c5b6 5be9      E..{..@.@.....[.
                ac10 70cf 0019 0403 e505 2e00 eb4d 975a      ..p.........M.Z
                5018 7fe0 a903 0000 3232 3020 6d61 7273      P.......220 mars
                2e61 766f 6361 646f 2e6e 6574 2053 656e      .avocado.net Sen
                646d 6169 6c20 342e 312f 534d 492d 342e      dmail 4.1/SMI-4.
                3120 7265 6164 7920 6174 204d 6f6e 2c20      1 ready at Mon,
                3120 4a75 6e20 3139 3938 2030 383a 3034      1 Jun 1998 08:04
                3a33 3020 2d30 3430 300d 0a                  :30 -0400..
08:04:27.689516 172.16.112.207.1027 > 197.182.91.233.smtp: . ack 84 win 32120 (DF)
                4500 0028 0275 4000 3f06 fadb ac10 70cf      E..(.u@.?.....p.
                c5b6 5be9 0403 0019 eb4d 975a e505 2e53      ..[......M.Z...S
                5010 7d78 59bf 0000 0000 0000 0000           P.}xY.........
08:04:27.703716 172.16.112.207.1027 > 197.182.91.233.smtp: P 1:26(25) ack 84 win 32120 (DF)
                4500 004l 0276 4000 3f06 facl ac10 70cf      E..A.v@.?.....p.
                c5b6 5be9 0403 0019 eb4d 975a e505 2e53      ..[......M.Z...S
                5018 7d78 4580 0000 4548 4c4f 2072 6f62      P.}xE...EHLO rob
                696e 2e65 7972 6965 2e61 662e 6d69 6c0d      in.eyrie.af.mil.
                0a                                           .
08:04:27.704112 197.182.91.233.smtp > 172.16.112.207.1027: P 84:110(26) ack 26 win 32736 (DF)
                4500 0042 0410 4000 4006 f826 c5b6 5be9      E..B..@.@..&..[.
                ac10 70cf 0019 0403 e505 2e53 eb4d 9773      ..p........S.M.s
                5018 7fe0 ecc5 0000 3530 3020 436f 6d6d      P.......500 Comm
                616e 6420 756e 7265 636f 676e 697a 6564      and unrecognized
                0d0a                                         ..
08:04:27.704764 172.16.112.207.1027 > 197.182.91.233.smtp: P 26:51(25) ack 110 win 32120 (DF)
                4500 004l 0277 4000 3f06 fac0 ac10 70cf      E..A.w@.?.....p.
                c5b6 5be9 0403 0019 eb4d 9773 e505 2e6d      ..[......M.s...m
                5018 7d78 4250 0000 4845 4c4f 2072 6f62      P.}xBP..HELO rob
                696e 2e65 7972 6965 2e61 662e 6d69 6c0d      in.eyrie.af.mil.
                0a                                           .
```

Figure 4.3: Printout of five network packets captured by TCPdump

network data consists of around 700,000 packets. The second problem is leveraging information from the packets. A single packet gives minimal information, however all of the packets from a connection between two computers provides all of the information exchanged during the connection. Therefore, it is necessary to analyze connections rather than packets. This enables an abstraction of information which will begin to provide a much more manageable data representation. A connection is simply a completed exchange of information between two computers. Connections can consist of less than ten packets or several hundred packets. The software utilized for the analysis of connections is TCPtrace, a TCP connection analysis tool written by Shawn Ostermann [116].

TCPtrace provides an overall summary of the connection, to include the number and type of packets, date-time group of the connection, duration, and other useful analysis. This is again represented in an alphanumeric format, however I wrote Perl program which parses the TCPtrace representation into a numeric representation which is compatible with machine learning software. An example of TCPtrace output for a TCPtrace connection is shown in figure 4.4.

After parsing TCPtrace output with the Perl program, the connection can be represented as a single tuple of 80 variables. Each tuple must furthermore be marked as either an attack or non-attack, and an example of how to mark tuples as

```
================================
TCP connection 21:
        host ao:       197.218.177.69:20
        host ap:       172.16.114.168:1206
        complete conn: yes
        first packet:  Wed Jun 10 08:00:46.552743 1998
        last packet:   Wed Jun 10 08:00:46.570673 1998
        elapsed time:  0:00:00.017930
        total packets: 12
        filename:      tcpdump
   ao->ap:                              ap->ao:
     total packets:             8           total packets:             4
     ack pkts sent:             7           ack pkts sent:             4
     pure acks sent:            2           pure acks sent:            2
     sack pkts sent:            0           sack pkts sent:            0
     dsack pkts sent:           0           dsack pkts sent:           0
     max sack blks/ack:         0           max sack blks/ack:         0
     unique bytes sent:      4763           unique bytes sent:         0
     actual data pkts:          4           actual data pkts:          0
     actual data bytes:      4763           actual data bytes:         0
     rexmt data pkts:           0           rexmt data pkts:           0
     rexmt data bytes:          0           rexmt data bytes:          0
     zwnd probe pkts:           0           zwnd probe pkts:           0
     zwnd probe bytes:          0           zwnd probe bytes:          0
     outoforder pkts:           0           outoforder pkts:           0
     pushed data pkts:          1           pushed data pkts:          0
     SYN/FIN pkts sent:       1/1           SYN/FIN pkts sent:       1/1
     urgent data pkts:          0 pkts      urgent data pkts:          0 pkts
     urgent data bytes:         0 bytes     urgent data bytes:         0 bytes
     mss requested:          1460 bytes     mss requested:          1460 bytes
     max segm size:          1460 bytes     max segm size:             0 bytes
     min segm size:           383 bytes     min segm size:             0 bytes
     avg segm size:          1190 bytes     avg segm size:             0 bytes
     max win adv:           32120 bytes     max win adv:           32736 bytes
     min win adv:             512 bytes     min win adv:           32120 bytes
     zero win adv:              0 times     zero win adv:              0 times
     avg win adv:           28169 bytes     avg win adv:           32428 bytes
     initial window:         2920 bytes     initial window:            0 bytes
     initial window:            2 pkts      initial window:            0 pkts
     ttl stream length:      4763 bytes     ttl stream length:         0 bytes
     missed data:               0 bytes     missed data:               0 bytes
     truncated data:            0 bytes     truncated data:            0 bytes
     truncated packets:         0 pkts      truncated packets:         0 pkts
     data xmit time:        0.013 secs      data xmit time:        0.000 secs
     idletime max:           11.7 ms        idletime max:           14.1 ms
     throughput:           265644 Bps       throughput:                0 Bps
================================
```

Figure 4.4: Printout of TCPtrace representation of a TCP connection

such is shown in chapter 8.

Bernhard Sick performed similar analysis based upon this same data set [76]. Sick examined feature selection techniques with a neural network to predict outcomes for this data. His data consists of 137 features (he uses a large number of binary variables - thus creating more variables). He shared this data and I have been able to apply both the one-class SVM and K-PLS as initial experiments. It is a goal in this research to utilize both the TCPtrace collected data and Sick's data to improve our understanding and ability to predict network based intrusions. A detailed explanation of a data flow diagram representing the data processing for both host based computer intrusion detection and network based computer intrusion detection is located in Appendix A.

CHAPTER 5
CREATING SYNERGY WITH UNSUPERVISED CLASSIFICATION

5.1 Introduction

This chapter includes exploratory work involving intelligent subspace selection and much of the original efforts that led to important findings regarding the fusion of classifiers included in chapter 6. A novel method for receiver operating characteristic (ROC) curve analysis and anomaly detection is proposed. The ROC curve provides a measure of effectiveness for binary classification problems, and this chapter specifically addresses unbalanced, unsupervised, binary classification problems. Furthermore, this work explores techniques in fusing decision values from classifiers and using ROC curves to illustrate the effectiveness of the fusion techniques. In describing an unbalanced classification problem, understand that this is a problem that has a low occurrence of the positive class (generally less than 10%). Since the problem is unsupervised, the one-class SVM is utilized. The chapter includes a discussion of the curse of dimensionality experienced with the one-class SVM, and to overcome this problem techniques involving subspace modeling are explored. For each subspace created, the one class SVM produces a decision value. The aggregation of the decision values occurs through the use of fuzzy logic, creating the fuzzy ROC curve. The primary source of data for this research is a host based computer intrusion detection dataset.

5.2 Introduction to the Problem

The purpose of this chapter is to illustrate synergistic combinations of multiple classifiers for the unbalanced, unsupervised binary classification problem. The SEA data and the Ionosphere data from the UCI data repository will be explored in this chapter.

Combinations of multiple classifiers (CMC), also referred to as multiple classi-

fication systems (MCS) or committee machines, is an active area of research today. Given the numerous classification techniques and vast problem area domain, research within this field seems only bounded by creativity and computational power. A particularly interesting problem that involves CMC is the unbalanced, unsupervised binary classification problem. Consider the following example that referred to as the "airport security problem". An airport security system exists in layers, all the way up to and including the aircraft while it is airborne. Each layer in this security system can be considered a classifier; the objective of each layer is to determine whether or not an intruder, for example, breaches security. How should these classifiers be arranged? How can the results of classifiers be combined to achieve synergistic results?

An unsupervised classifier is handicapped by the fact that it often cannot learn from true positive (intruders) examples. The only examples available to learn from are true negatives (non-intruders). Furthermore, the problem is unbalanced. This means that the frequency of intruders is very small; in an airport, this number of true positives is a tiny fraction of a percent. Some airports work for years without experiencing an intruder. Yet the cost of not identifying a true positive can be catastrophic, and the cost of falsely identifying non-intruders, or having too many false positives, can create is also formidable.

Given a classification problem of high dimension (perhaps >10 variables), initial experimental results indicate that the creation of subspaces and aggregation of the subspace classification decision values results in improved classification over a model that utilizes all variables at once (the unsupervised classification model that will be utilized is the one-class SVM [127]). This is often known as the "curse of dimensionality". Creating subspaces for outlier detection is not a new concept [74]. However, considering this problem as a function of one-class SVM outputs to create a "fuzzy ROC curve" is novel. As multiple classifiers combine, synergistic improvement in ROC curves can be observed. Fuzzy logic is the basis for the aggregation. Each classifier will create a decision value that ranges from -1 to +1, where +1 should indicate the negative (non-intruder) class, and -1 indicates the positive (intruder) class. These decision values represent a degree of membership in an intruder or a

non-intruder class. The decision values must be combined to make a final decision, and fundamentals of fuzzy logic can be used to aggregate these decision values.

As mentioned earlier, unsupervised learning does not perform well in higher dimensions. A valid question would be to ask why not try a dimension reduction technique, such as principal components. Principal components work well with balanced classification problems, however caution is necessary with unbalanced problems. Often the difference between an intruder and non-intruder is subtle, and principle components can dilute the information content of the variables.

The overall performance measure is the area under the ROC curve, or AUC, with a secondary performance measure of good performance at low false positive rates. Overall, the ROC curve created from the subsets should exceed the curve created by processing one lump sum of variables. Claiming success requires this to occur. The results section shows several instances where this occurred. Secondly, in the range of low false positives, where intrusion detection and security systems typically operate, performance must be good which means that in the range from 0 to 50% true positive, the false positive rate diverges slowly from 0. Examples of this are shown in the results section. Typically good low false positive range performance and increased AUC will occur simultaneously, but not always.

Figure 5.1 is a cartoon sketch that summarizes the goal of the work in this chapter. There is an assumed high dimensional dataset, $\mathbf{X} \in \mathbb{R}^{N \times m}$. $\mathbf{X}$ contains N instances or observations, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, where $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$. Modeling the data in subspaces reduces dimensionality problems and fusing the results of subspace models creates robust results.

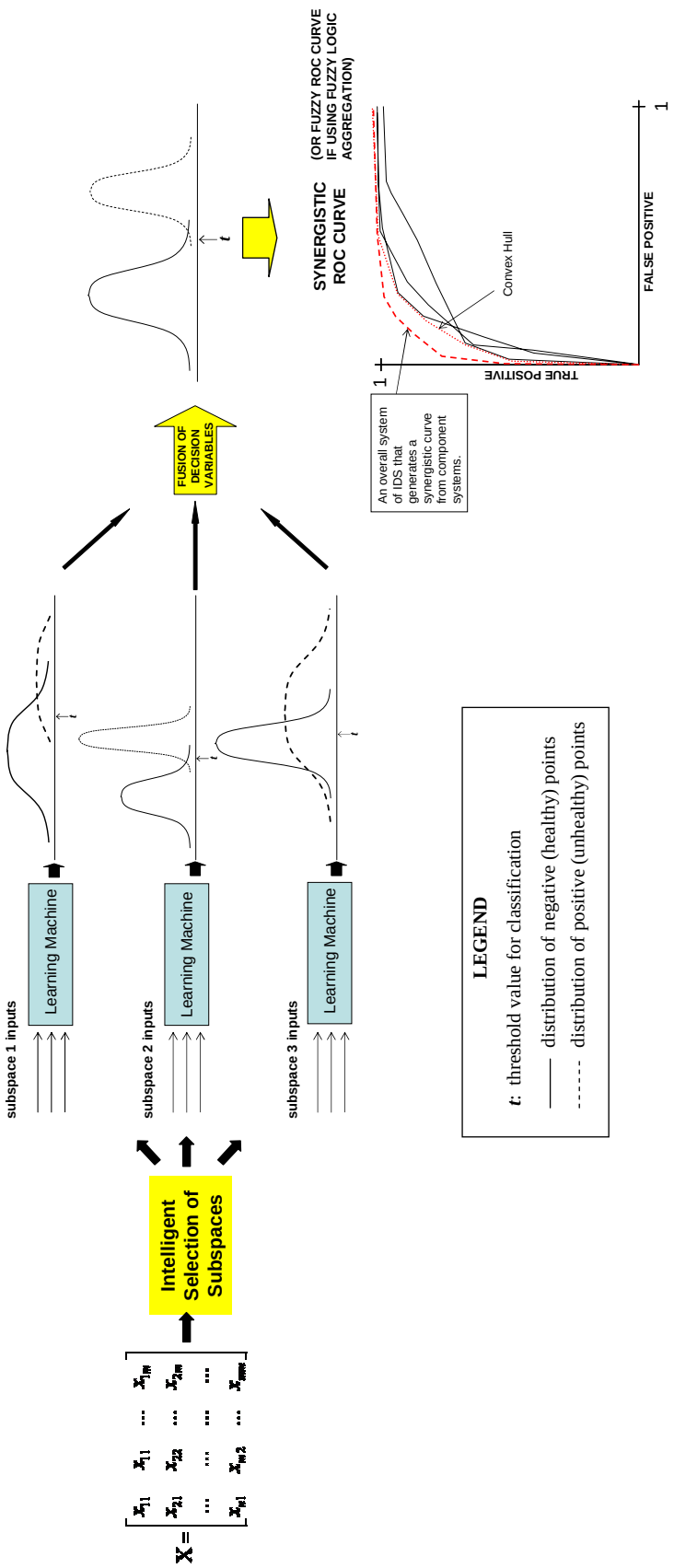

Figure 5.1: A sketch of subspace modeling to seek synergistic results

5.3 Dataset and Model

This chapter explores many facets of recent research, to include intrusion detection models, outlier detection, and fusion of classifiers using fuzzy ROC curves. Recent work with classifier fusion and fuzzy ROC curves will be discussed during the presentation of the methods for that material.

Schonlau et. al. [41–43, 128, 129] conducted the original work with the data explored in this chapter. A detailed discussion of this data can be found in sections 2.5.1 and 4.3. Throughout these experiments normalization of the data occurs(by subtracting the mean and dividing by the standard deviation). The dataset contains 5000 observations and a host of variables to measure these observations (see [50, 135] and section 4.3 for a description of variables). The training set consists of 2500 observations and the test set contains 2500 observations. After eliminating all positive cases from the training data, 2391 negative cases remain which are used for training the one-class SVM. Recall that the one-class SVM trains with solely negative instances. In the testing data, there are 122 positive cases out of the 2500 observations.

The one-class SVM is an outlier detection technique originally proposed in [127]. Stolfo and Wang [132] successfully apply the one-class SVM to this dataset and compare it with several of the techniques mentioned above. Chen uses the one-class SVM for image retrieval [27]. The simplest way to express the one-class SVM is to envision a sphere or ball, and the object is to squeeze all of the training data into the tightest ball feasible. This is analogous to the idea of variance reduction for distribution estimation; given a set of data, I want to estimate a distribution that tightly defines this data, and any other data that does not look the same will not fit this distribution. In other words, once the distribution is estimated, data that does not fit the distribution will be considered an outlier or not a member of the negative class. Consider the following formulation of the one-class SVM originally from [127] and also clearly explained in [27]:

If we consider $X_1, X_2, ..., X_l \in \chi$ instances of training observations, and Φ is a mapping into the feature space, F, from χ.

$$\min_{R\in\mathbb{R},\zeta\in\mathbb{R}^l,c\in F} \quad R^2 + \frac{1}{vl}\sum_i \zeta_i$$

$$\text{subject to} \quad \| \Phi(X_i) - c \|^2 \leq R^2 + \zeta_i, \quad \zeta_i \geq 0 \text{ for } i \in [l] \tag{5.1}$$

This minimization function attempts to squeeze R, which can be thought of as the radius of a ball, as small as possible in order to fit all of the training samples. If a training sample will not fit, ζ_i is a slack variable to allow for this. A free parameter, v, enables the modeler to adjust the impact of the slack variables. The output, or decision value for a one-class SVM, takes on a values from -1 to +1, where values close to +1 indicate datapoints that fit into the ball and values of -1 indicate datapoints lying outside of the ball. A detailed discussion of the one class SVM can be found in section 2.2.3.2.

5.3.1 Curse of Dimensionality

It is commonly understood that high-dimensional data suffer from a curse of dimensionality. This curse of dimensionality involves the inability to distinguish distances between points because as dimensionality increases, every point tends to become equidistant as volume grows exponentially. This same curse of dimensionality occurs in the one-class SVM. (The SVM tool used for this research is LIBSVM [24] with a linear kernel.)

Figure B.4 illustrates an experimental example of the curse of dimensionality where there are originally 27 meaningful variables, however meaningless probe variables (uniform (0,1) random variables) are added to create degradation. The area under the ROC curve, or AUC, will serve as a measure of classifier performance. See chapter 3 for a discussion of ROC curves. Tom Fawcett also provides an excellent discussion of ROC curves in [55].

5.4 Method to Create Fuzzy ROC Curves

The following technique seeks to overcome this curse of dimensionality. The technique involves creating subspaces of the variables and aggregating the outputs of the one-class SVM for each of these subspaces.

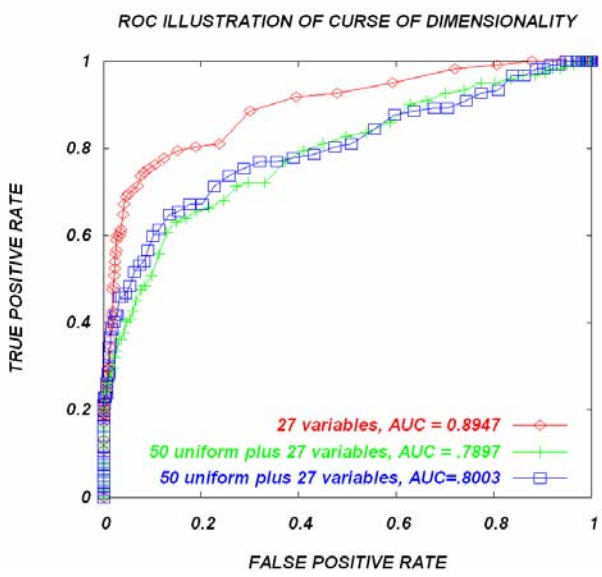

Figure 5.2: Curse of dimensionality induced by the introduction of probe variables

5.4.1 Subspace Modeling

Intelligent subspace modeling is an important first step. Orthogonal subspaces are desired, because we are interested in subspaces that measure different aspects of the data. The idea of creating diverse classifiers is not novel [15, 96, 97, 99], however in the literature the measures of classifier diversity involve functions of the classifier output. This is feasible with supervised learning, however in unsupervised learning this is more difficult because there are no true positive examples to measure diversity against. I propose measuring diversity through the actual data. The following method involves an analysis of the correlation between principal components of each subspace.

Given a Mahalanobis scaled data matrix $\mathbf{X}$, containing m variables that measure n observations, create l mutually exclusive subspaces from the m variables. Assume there are k variables in every subspace if m is divisible by l. Experience with the one-class SVM indicates that for $k > 7$, increased dimensionality begins to degrade performance, however this is simply a heuristic and may vary depending upon the unsupervised classifier selected. For each subspace, principal components can be calculated. $\mathbf{L}$ will refer to the matrix that contains the principal component loading vectors (eigenvectors). To determine correlation between principal compo-

nents, calculate the principal component scores for each subspace, where **S**=**XL**. Let π_i represent subspace i, and consider $\mathbf{S}_i$ as the score matrix for the π_i. Calculate the pairwise comparison for every column vector in $\mathbf{S}_i$ against every column vector in $\mathbf{S}_j$, $i \neq j$. This would be the equivalent of concatenating $\mathbf{S}_i$ for all i and calculating the correlation matrix, Σ.

We are interested in values approaching zero for every pairwise correlation across subspaces (principal components within subspaces are orthogonal and therefore their correlation is zero, as seen in Figure 5.5). However, there are a combinatoric number of subspace combinations to explore.

$$\text{Number of subspace combinations} = \begin{pmatrix} m \\ k \end{pmatrix} \begin{pmatrix} m-k \\ k \end{pmatrix} ... \begin{pmatrix} 2k \\ k \end{pmatrix} \tag{5.2}$$

Equation 5.2 assumes that m is divisible by l, and even if this is not true the equation is almost identical and on the same order of magnitude. Our approach to search this subspace involved the implementation of a simple genetic algorithm, utilizing a chromosome with m distinct integer elements representing each variable. There are many possible objective functions that could pursue minimizing principal component correlation between subspaces, and I utilized the following letting $q \in (1, 2, ..., l)$:

$$\min \max_{\forall \pi_q} \mid \rho_{ij} \mid \quad \forall (i \neq j) \tag{5.3}$$

The fitness of each member is simply the maximum $\mid \rho_{ij} \mid$ value from the correlation matrix such that ρ_{ij} measures two principal components that are not in the same subspace.

5.4.1.1 The Genetic Algorithm

The purpose of this GA is to seek optimality for equation 5.3. Appendix C contains the perl code written to implement this GA. Principal components from the same subspace will have 0 correlation, however principal components from different subspaces will have correlation. If principal components are correlated, then this is

an indicator that the subspaces measure the same behavior. Therefore, uncorrelated principal components are desirable.

The chromosome consisted of 26 elements, and each of these elements represented a variable. The initial population consisted of 50 randomly ordered chromosomes. Figure 5.3 is an example of the chromosome encoding used in this algorithm(this is actually the best configuration discovered to date). The three subsets are extracted as shown, and then the $Analyze^{TM}$ [46] software package calculates the principal components and correlation of the principal components across subspaces.

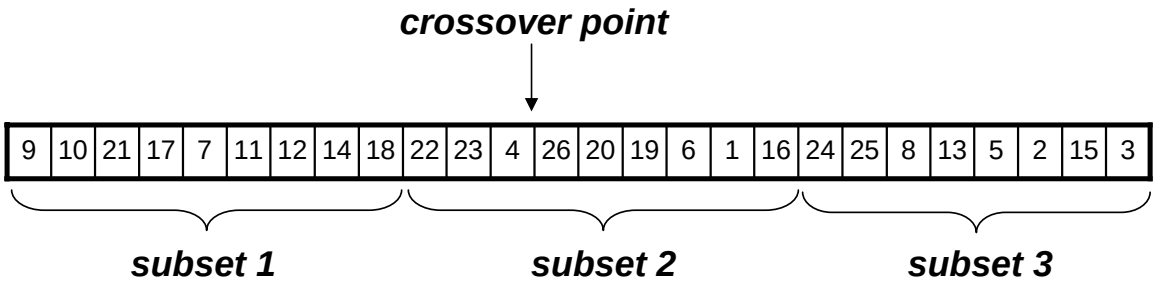

Figure 5.3: Illustration of chromosome and subspaces

The selection of chromosomes followed the common roulette wheel process, where chromosomes with a better fitness receive a higher probability of being selected. In order to retain the fittest chromosome found, an elitist rule retains the single best chromosome and passes it to the next generation. Immigrants were also implemented. These were two new random chromosomes who took the place of the least fit chromosomes selected for breeding. After selecting a group of chromosomes for breeding, these chromosomes had a 40% chance of directly passing to the next generation and a 60% chance of crossover. Crossover occurred at the point shown in Figure 5.3. This crossover option was somewhat unorthodox since a clean crossover could not occur due to the fact that every element in the chromosome must be unique. Therefore, after executing a crossover, the new chromosomes required adjustment to ensure unique elements. This adjustment occurred only on the first half of the new chromosome (before the crossover point); ie, if a redundant element occurred in the new chromosome, this would randomly be replaced by one of the variables not yet accounted for in the new chromosome.

The last step involved mutation. Mutating chromosomes also required special

consideration due to constraints of chromosome elements. Each element had a 1% chance of encountering mutation. If an element was selected for mutation, the element was simply swapped with its mirror element. If the chromosome contains l elements, and the k^{th} element is selected for mutation, then I swap this with the $(l-k)^{th}$ element, which I refer to as its mirror.

5.4.2 Output Processing

After selecting the subspaces, prediction modeling begins. As mentioned previously, the choice for a prediction model is the one-class SVM with a linear kernel. However, the problem of classifier fusion still persists. Classifier fusion techniques have been discussed in [15, 96, 97, 99]. Classifier fusion is a relatively new field and it is often criticized for lack of theoretical framework and too many heuristics [97]. The methods in this chapter do not claim to provide a solution to this criticism. The presented method of classifier fusion is a blend of techniques from fuzzy logic and classifier fusion, and although it may be considered another heuristic, it is operational and should generalize to other security problems.

5.4.2.1 Mapping into Comparable Decision Spaces

For each observation within each subspace selected, the classifier will produce a decision value, d_{ij}, where d_{ij} represents the decision value from the j^{th} classifier for the i^{th} observation. Since the distribution of the output from almost any classification technique is questionable, first consider a nonparametric measure for the decision value, a simple ranking. o_{ij} represents the ordinal position of d_{ij} (for the same classifier, meaning j remains constant). For example, if d_{71} is the smallest value for the 1^{st} classifier, $o_{71} = 1$. This nonparametric measure allows comparison of classifiers without considering the distribution. However, do not rule out the distribution altogether. Next create p_{ij}, which is the Mahalanobis scaled (normalized) value for d_{ij}. In order to incorporate fuzzy logic, o_{ij} and p_{ij} must be mapped into a new space of real numbers, let us call Λ, where $\Lambda \in (0, 1)$. This mapping will be $p_{ij} \rightarrow \delta_{ij}$ and $o_{ij} \rightarrow \theta_{ij}$ such that $\delta_{ij}, \theta_{ij} \in \Lambda$. For $o_{ij} \rightarrow \theta_{ij}$ this is a simple scaling procedure where all o_{ij} are divided by the number of observations, m, such that $\theta_{ij} = o_{ij}/m$. For $p_{ij} \rightarrow \delta_{ij}$, all p_{ij} values < -1 become -1, all p_{ij} values > 1 become

1, and from this point $\delta_{ij} = (p_{ij} + 1)/2$.

5.4.2.2 Fuzzy Logic and Decisions with Contention

There are now twice as many decision values for every observation as there were numbers of classifiers. Utilizing fuzzy logic theory, T-conorms and T-norms can be considered for fusion. The choice between T-norms and T-conorms depends upon the type of decision. The medical community is cautious of false negative tests, meaning that they would rather have error on the side of falsely telling someone that they have cancer as opposed to letting it go undetected. The intrusion detection community is concerned about minimizing false positives, because too many false positives render an intrusion detection system useless as analysts slog through countless false alarms. In the realm of one-class SVMs, the original decision values will take on values ranging generally from -1 to +1, where values closer to +1 indicate observations that fit inside the ball or estimated distribution (indicating non-intruders), and values closer to -1 indicate outliers (potential intruders). Consider the max and min, simple examples of a respective T-conorm and T-norm. Systems that need to be cautious against false negatives will operate in the realm of the T-norms, creating more false alarms but missing fewer true positives. Systems that need to be cautious against false negatives will operate in the realm of the T-conorms, perhaps missing a few true positives but generating fewer false positives. Figure 5.4 illustrates the domain of aggregation operators.

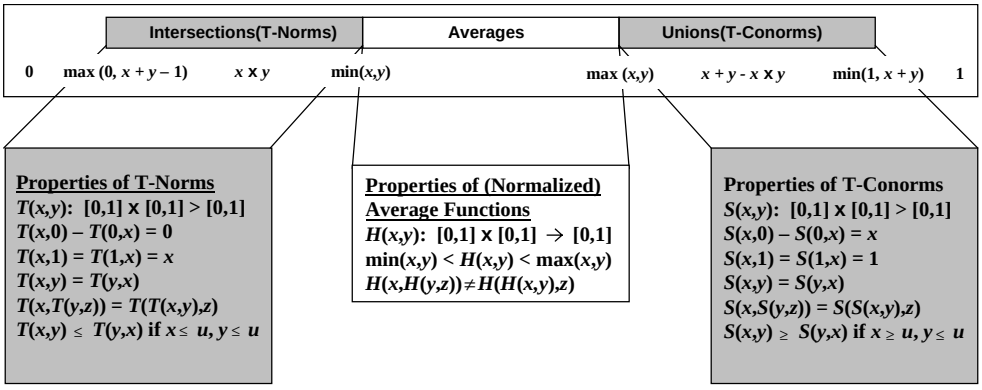

Figure 5.4: Fuzzy aggregation operators

One problem with T-norms and T-conorms is that contention within aggrega-

tion is not captured. By contention I am referring to a vast difference of decision values between classifiers. However, contention can be captured and considered appropriately. Typically, if contention exists, a system needs to reflect caution. In other words, if we are minimizing false positives and contention exists in a decision, we may simply choose negative or choose a different aggregator for contentious decisions. If contention exists in a medical decision, it is likely that the initial diagnosis will report positive (cancer detected) and then further tests will be pursued. There are numerous ways to measure contention, and one of the simplest is to consider the difference between the max and min decision values. If this difference exceeds a threshold, contention exists and it may be best to choose a different aggregator or make a cautious decision.

5.5 Results with Masquerading Data

Experimental results involved the SEA dataset. There are m=26 variables and n=2500 observations in the training data. For our subspace selection, there are l=3 subspaces creating subspaces containing 9, 9, and 8 variables respectively. For each subspace we consider three principal components. The genetic algorithm used the fitness function shown in Equation 5.3, roulette wheel selection, a crossover rate of .6 and mutation rate of .01. Our number of generations = population size = 50. Figure 5.5 is the correlation matrix of the principal components from our selected subspaces, where $\max_{\forall \pi_q} \mid \rho_{ij} \mid = .4 \; \forall (i \neq j)$.

Given this subspace configuration, LIBSVM can calculate the one-class SVM decision variables. Given the decision variables d_{ij}, map $d_{ij} \rightarrow o_{ij} \rightarrow \theta_{ij}$ and $d_{ij} \rightarrow p_{ij} \rightarrow \delta_{ij}$ as described in section 5.7. The decision rule was to take the maximum value unless there was contention $> .5$, and in this case take the median of all decision values. The ROC curves shown in figure 5.6 and tabled values in Table 5.2 illustrate the results. The red (superior) ROC curve represents the result with some type of aggregation, the green (inferior) ROC curve represents the result without any aggregation and utilizing 26 variables for the one-class SVM input.

As Figure 5.6 illustrates, different decision rules create various outcomes. The best plot, (d), integrates all decision variables as the algebraic sum. Table 5.1

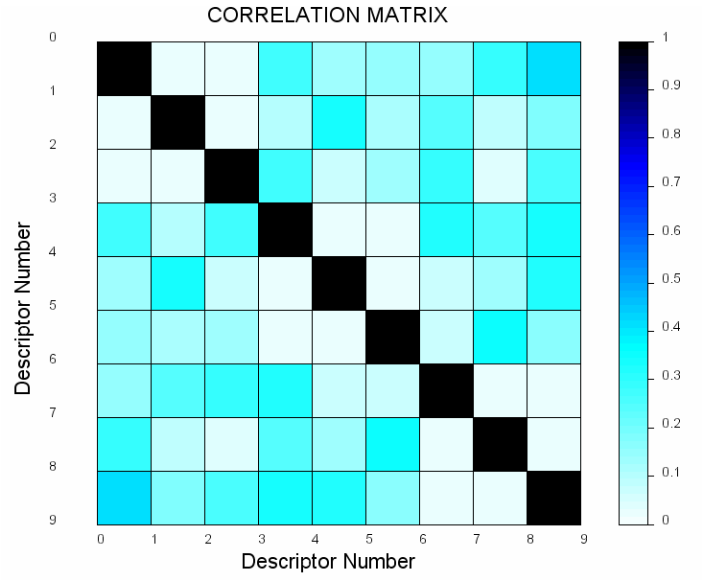

Figure 5.5: Correlation matrix of subspace principal components

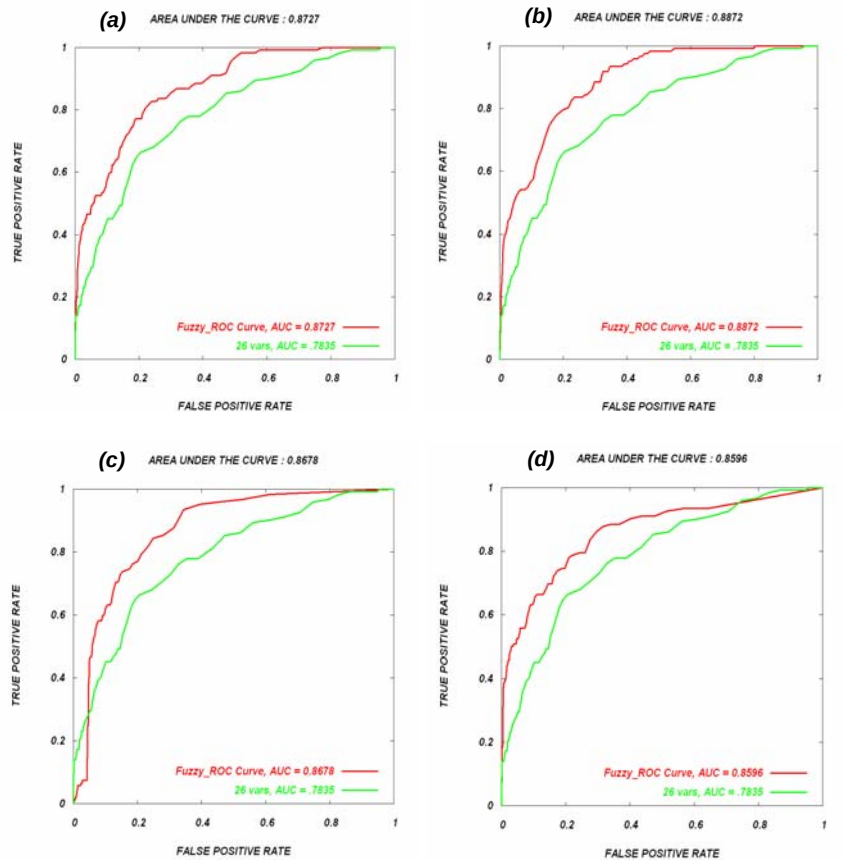

Figure 5.6: ROC plots illustrating effect of different decision rules

Table 5.1: Decision rules for ROC plots in Figure 5.6

ROC Plot	**Decision Rule for Each Observation (i); (t = threshold for contention)**
(a)	algebraic sum of $(\theta_{ij})\forall j$
(b)	algebraic product of $(\theta_{ij})\forall j$
(c)	if $t < .5$, algebraic sum of $(\delta_{ij}, \theta_{ij})\forall j$; if $t \geq .5$, median $(\delta_{ij}, \theta_{ij})\forall j$
(d)	algebraic sum of $(\delta_{ij}, \theta_{ij})\forall j$

Table 5.2: Overall Results with Best Subsets.

	$\forall\delta_{ij}, \theta_{ij}$		$\forall\theta_{ij}$		$\forall\delta_{ij}$	
	AUC	comment	AUC	comment	AUC	comment
T-norms						
minimum	.695		.8632	poor low FP	.7	
algebraic product	.5204		.887	low FP not the best	.676	
minimum with contention	.6907		.8458		.6929	
algebraic product with contention	.4999		.8363	poor high FP	.677	
T-conorms						
maximum	.8206	poor high FP	.7967		.825	
algebraic sum	.8596	best low FP	.8727		.858	
maximum with contention	.849		.8105		.86	
algebraic sum with contention	.8678	poor low FP	.8548		.887	low FP not the best

describes the decision rules.

These results provide interesting insight. It is an overall assumption that T-conorms provide the best rules for the security classification problem. However, for the ordinal decision variables (θ_{ij}), T-norms performed well. The ordinal variables are interesting in themselves since there is no distance considered when comparing variables; the only aspect considered is order. This is a non-parametric method for comparison, and this is done since the distribution of the decision variables is not easy to estimate and perhaps should not be assumed. The distance based decision

variables (δ_{ij}) do not perform well with T-norms. Overall, the ROC plot d appears to perform best since there is exceptional performance for low false positive rates and the overall AUC is among the best. The idea of contention deserves much more study, and this parameter is likely to need tuning. It obviously does impact the overall results and warrants consideration, however it is not entirely understood. Table 5.3 is an illustration of poor results, which should reinforce some of the claims previously made.

Table 5.3: Overall Results with Worst Subsets.

	AUC $\forall\delta_{ij}, \theta_{ij}$	AUC $\forall\theta_{ij}$	AUC $\forall\delta_{ij}$
T-norms			
min	.6167	.7767	.617
algebraic product	.5204	.775	.62
min with contention	.726	.787	.648
algebraic product with contention	.557	.790	.653
T-conorms			
max	.7457	.769	.7466
algebraic sum	.814	.775	.818
max with contention	.6874	.769	.6929
algebraic sum with contention	.7563	.725	.762

Table 5.3 illustrates the results obtained for the worst subset encountered (the configuration that scored highest for our minimization fitness function). This is interesting to observe since it is difficult to show that some type of optimality has been reached by minimizing the correlation between principal components. Empirically, this shows that there are much worse configurations and it is worth exploring for improved subspace configurations.

5.6 Seeking Diverse Subspaces with Nonparametric Statistics

It is widely known that diversity is desirable for classifier fusion [97, 98], and there are numerous methods for measuring classifier diversity for supervised classification. It is undesirable to have multiple classifiers that all make the same errors in the same direction; it is desirable to have classifiers making different mistakes on different instances, and when combined, synergistic performance occurs through intelligent combination. A simple regression model illustrates this idea. This model can be expressed as $\mathbf{Xw}=\mathbf{y}$, where $\mathbf{X} \in \mathbb{R}^{N \times m}$, $\mathbf{w} \in \mathbb{R}^{m \times 1}$, $\mathbf{y} \in \mathbb{R}^{N \times 1}$. If $\mathbf{y}$ is known for the training data, the measures of diversity shown in [97, 98] apply. However, when $\mathbf{y}$ is either unknown or only contains the negative class, measures of diversity must involve $\mathbf{X}$. Therefore, since our desire is to create subspaces of a high dimensional dataset, we seek diverse subspaces.

In order to measure this diversity, a distance measurement, d_{ij}, will be calculated for the i^{th} observation in every j^{th} subspace. The distance measurement is the Euclidean distance of the observation point to the subspace centroid. However, other distance measurements should not be discounted, and nonlinear kernel distance measurements can also be considered. This distance measurement will provide the basis of our intelligent subspace modeling. Our interest is to find subspaces that are not correlated with respect to d_{ij}. If subspaces are correlated with respect to d_{ij}, these subspaces capture similar behavior. Uncorrelated subspaces indicate subspaces that are somewhat orthogonal, and we interpret this as diversity.

Figure 5.7 illustrates the idea of correlated versus uncorrelated subspaces based upon Kendall's W, a nonparametric measure of concordance which calculates agreement across multiple subspaces using only ranks. The vertical axis in figure 5.7 measures d_{ij}, and the ranking of the points is obvious from plots. Similar ordering occurs in the subspaces on the left with a higher W, however the subspaces on the right contain a much more random order, and W reflects a lower value.

Kendall's W provides a scalar measurement of agreement or disagreement of the rankings between subspaces. In order to compute W, first map $d_{ij} \rightarrow R_{ij}$ such that R_{ij} represents the rank of the i^{th} point distance-wise with respect to the j^{th}

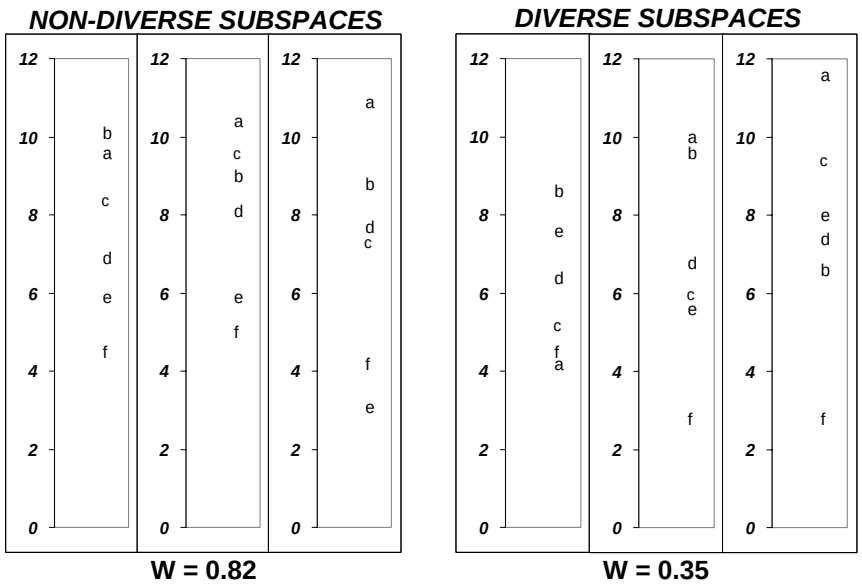

Figure 5.7: A comparison of correlated and uncorrelated subspaces

centroid. Assuming that there are N points in our training data and l subspaces, Kendall defines W as follows in [89]:

$$W = \frac{12S}{l^2(N^3 - N)} \tag{5.4}$$

$$\text{where } S = \sum_{i=1}^{N} \left(\left(\sum_{j=1}^{l} R_{ij} \right) - \frac{l(N+1)}{2} \right)^2$$

It is our hypothesis that diverse subspaces will create a small W, and this diversity will provide improved ensembles for unsupervised learning. The same genetic algorithm is used to minimize W.

Receiver operating characteristic (ROC) curves provide an elegant, simple representation of the performance of a binary classification system, based entirely upon nonparametric ranks. The curve presents the relationship between a false positive rate and a true positive rate across the full spectrum of operating points, which can also be considered the full spectrum of a continuous threshold value for a decision method. The nonparametric interpretation of the ROC curve and the AUC is discussed in [17, 71, 109]. Many machine learning algorithms provide a real number, or soft decision value, as the output. It is difficult, and often not necessary, to as-

sociate this output to a probability distribution. It is more meaningful to consider this output as a degree of confidence towards one class or the other. Interpreting this soft decision value as a ranking leads directly to the nonparametric statistics associated with the area under the ROC curve. The Wilcoxon Rank Sum statistic, which is directly proportional to the Mann-Whitney U statistic, provides this association. Chapter 3 provides a detailed summary of ROC curve construction and the relationship between ROC curves and ranks.

5.7 Mapping into Comparable Decision Spaces

For each observation within each subspace selected, the classifier will produce a decision value, D_{ij}, where D_{ij} represents the decision value from the j^{th} classifier for the i^{th} observation. o_{ij} represents the ordinal position, or rank, of D_{ij} (for the same classifier, meaning j remains constant). For example, if D_{71} is the smallest value for the 1^{st} classifier, $o_{71} = 1$. In order to incorporate fuzzy logic, o_{ij} must be mapped into a new space of real numbers, let us call Λ, where $\Lambda \in (0, 1)$. This mapping will be $o_{ij} \rightarrow \theta_{ij}$ such that $\theta_{ij} \in \Lambda$. For $o_{ij} \rightarrow \theta_{ij}$ this is a scaling procedure where all o_{ij} are divided by the number of observations, N, such that $\theta_{ij} = o_{ij}/N$.

5.8 Experimental Results

Two datasets have been explored for the experimental results. The first dataset, which we will refer to as the Schonlau et. al. or SEA dataset, is a computer intrusion dataset originally created by Schonlau and discussed in [41–43, 128, 129]. The data consists of the combination of text mining variables (described in [50]) and recursive data mining variables (described in [135]) derived from the SEA dataset. In total there are 26 variables.

The results shown in table 5.4 and figure 5.8 illustrate the improvements obtained through our nonparametric ensemble technique for unsupervised learning. The plot of the ROC curves shows the results from using 26 original variables that represented the SEA data as one group of variables with the one-class SVM and the result of creating $l = 3$ subspaces of features and fusing the results to create the fuzzy ROC curve. It is interesting to notice in the table of results that nearly

every aggregation technique demonstrated improvement, with the most significant improvement in the T-norms.

Table 5.4: Results of SEA data with diverse and non-diverse subsets

	DIVERSE (AUC)	NON-DIVERSE (AUC)
T-norms		
minimum	.90	.84
algebraic product	.91	.85
minimum with contention	.86	.81
algebraic product with contention	.93	.90
T-conorms		
maximum	.84	.80
algebraic sum	.89	.85
maximum with contention	.82	.77
algebraic sum with contention	.86	.81

for contention, $t = .5$

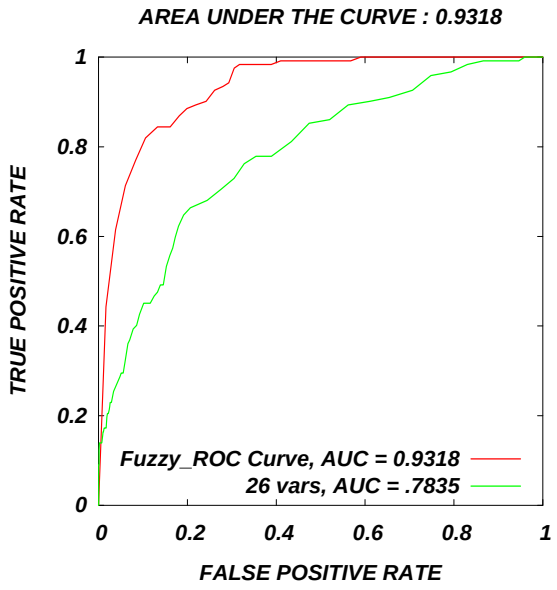

Figure 5.8: ROC for SEA data using algebraic product with contention

The ionosphere data is included to illustrate the performance of our ensemble technique with a balanced benchmark data set. This dataset is available from the

UCI repository, and it consists of 34 variables that represent different radar signals received while investigating the ionosphere for either good or bad structure. For this experiment we again chose $l = 3$.

Table 5.5: Overall results of ionosphere data with diverse and non-diverse subsets

	DIVERSE	NON-DIVERSE
T-norms		
minimum	.96	.953
algebraic product	.61	.64
minimum with contention	.86	.92
algebraic product with contention	.85	.91
T-conorms		
maximum	.69	.70
algebraic sum	.69	.70
maximum with contention	.82	.88
algebraic sum with contention	.85	.91

for contention, $t = .5$

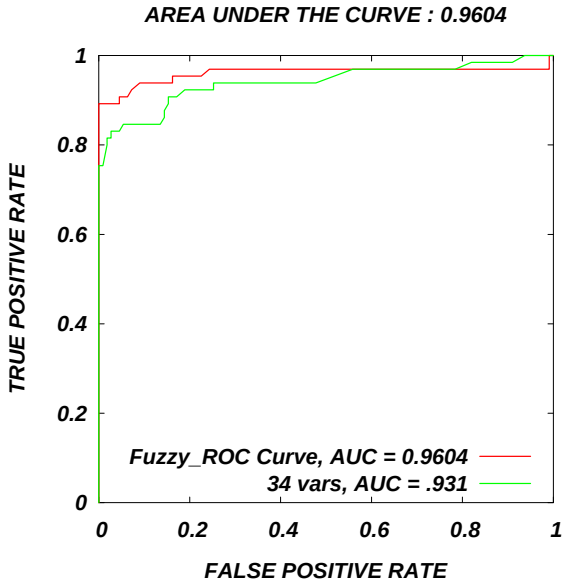

Figure 5.9: ROC plot for ionosphere data with minimize aggregation technique

It is very logical to ask why a simple dimension reduction technique, such

as principal components, is not sufficient to overcome the curse of dimensionality. Principal components capture variance, and by utilizing principal components the modeler is assuming that the variables from the training and testing data follow similar distributions, and furthermore that it is possible to identify novelties as an outlier along one of these axis of maximal variance, or principal component. If this novelty occurs as an outlier from only one variable, and this variable contributes minimal variance to the dataset in the training sample, it is likely that this novelty will go undetected. Furthermore, it is important that subspaces consist of meaningful dimensions for causality analysis [117].

For both the SEA data and the ionosphere data, principal components analysis was used as a dimension reduction technique to compare performance versus the ensemble method. For the SEA data, the PCA technique did improve classification reaching an AUC of .91, however the diverse subspaces with T-norm aggregation performed comparable to this and in some cases better. For the ionosphere data, however, the PCA technique actually degraded performance, achieving an AUC of 0.89 and falling well short of the best ensemble shown above.

T-norms provide the best aggregation for both datasets. This illustrates the idea of diversity, where different classifiers apparently correctly identify different positive cases. Every aggregator does not create improved performance with the ionosphere data; actually, it is only the minimization operator that improves performance. Furthermore, the impact of diversity does not seem as significant with the ionosphere data as it is with the Schonlau data. It is possible that a different distance measure could be appropriate for the ionosphere data.

The purpose of this chapter was to illustrate the work that took place during the course of this research to explore subspace modeling and model aggregation. Throughout the chapter, T-norms performed well. The next chapter will describe why a particular T-norm, the min, consistently performs well with unbalanced data. The important findings included in the next chapter would never have been possible without the exploratory work discussed in this chapter.

CHAPTER 6
SYNERGISTIC CLASSIFIER FUSION

This chapter includes the most important findings of this thesis. Unbalanced classification is a fundamental theme of this thesis, and this chapter explains why model fusion for unbalanced datasets performs differently than model fusion for balanced data. Exposing this difference provides researchers with an additional parameter, the balance of the data, which must be considered when building ensembles of classification models.

6.1 Introduction

Consider several sensors responsible for detecting some type of anomolous behavior. The sensors serve as sentries to a larger system. Suppose that every sensor reacts to every observation, evaluating or ranking the observation based upon a history of known behavior. Suppose that for each observation, some of the sensors have an opportunity to closely observe and measure an observation (a "good" measurement), and some of the sensors remotely observe (a "poor" measurement). All of the sensor measurements will be considered for the decision, and it is unknown which sensors closely observe and which ones remotely observe. How should the measurements of these sensors fuse to create the best signal? What other considerations regarding the observed population should be included when choosing the fusion method? What other information should be considered when choosing the fusion method for this situation?

These are a few of the questions that this chapter addresses. This scenario is also one potential application of synergistic classifier fusion. Synergistic classifier fusion is an ensemble technique designed for the unbalanced classification problem, very applicable with SCPs. Synergistic classifier fusion uses one additional piece of information to improve performance - assumed imbalance of the classes. With this simple assumption, it is possible to take advantage of the behavior of rank distributions and use min or max aggregators for synergistic performance. The generic

framework of this chapter considers the classic case of model ensembles. With model ensembles, fusion of ranks is all the more important because the underlying distributions of the model decision values is unknown. Several important novelties stem from this chapter:

- Pseudo ROC curves. ROC curves are a performance metric. However, it is entirely possible to examine the underlying statistics which create ROC curves to better understand classifier behavior. Typically ROC curves are built from a classification scenario and simply observed for what they are. However, how does an ROC curve with an AUC of 0.7 differ from an ROC curve with an AUC of 0.9? How does an ROC curve with an AUC of 0.9 that measures a classification problem with a 90% negative class differ from an ROC curve with an AUC of 0.9 that measures a classification problem with a 50% negative class? These questions can be explored with Pseudo ROC curves.

- Rank distributions from pseudo ROC curves. Rank distributions illustrate the behavior of classifiers from a non-parametric position. ROC curves are non-parametric, and the underlying distributions of the ranks which create ROC curves are non-parametric. When comparing two classifiers, comparing them with non-parametric statistics makes sense. The underlying distribution of classifier decision values is unknown - using (non-parametric) ranks enables comparison of classifiers on a level field. These rank distributions also lead to consideration of the max and min aggregation or fusion metrics.

- The min and max aggregators provide robust classifier fusion for imbalanced classification problems. This chapter will discuss the behavior of rank distributions for imbalanced classification problems - rank distributions of imbalanced classification problems behave with different likelihoods (discrete rank probabilities) than a balanced problem. It is possible to take advantage of this difference in likelihoods to improve classification.

This chapter discusses the methods utilized to achieve synergistic classifier fusion and the underlying theory explaining why this synergistic classifier fusion

occurs. The fusion methods described provide consistently robust solutions to the security classification problem. The first part of this chapter introduces the fusion method and the experimental results achieved. The second part of this chapter explains why this technique consistently performs well.

6.2 Recent Related Work

Recent work involving ensemble methods which contain combinations of feature subsets include Breiman's work on Random Forests [19] and Ho's work with Random Subspaces [74]. Both of these methods use a random approach for subspace selection, which is different from the method proposed in this chapter. Furthermore, both of these authors utilize decision trees and the average (avg) function to fuse or aggregate the ensemble. This chapter will show that the avg aggregator alone is not the best aggregator for the security classification problem. Furthermore, the methods described in this chapter generalize to the machine learning domain where it is assumed that model output involves real valued decision values as opposed to a binary decision value which is the case with decision trees.

Other recent work in ensemble techniques include Kuncheva's extensive work on combinations of pattern classifiers and diversity of classifier combinations in [96–99]. Bonissone et. al. and Evangelista et. al. have introduced the idea of fuzzy fusion of classifiers in [15] and [48, 49], respectively. Some of the work in cluster ensembles ([133], [117]) also relates to the material in this chapter. The pseudo ROC curve and rank distributions discussion in this chapter has also been published by Evangelista et. al. in [51].

6.3 Pseudo-ROC Curves

A study of pseudo-ROC curves and rank distributions will provide support and insight to the underlying behavior of classifier fusion for the security classification problem. This discussion is critical in understanding why certain classifier fusion metrics work best when fusing multiple models in the security classification domain.

ROC curves are based entirely upon ranks. Furthermore, the Mann-Whitney U statistic, which is equivalent to the area under the ROC curve, is also equivalent to

the probability that any random positive instance is ranked higher than a negative example. Stated formally, let us refer to $R(\mathbf{x}_i)$ as the rank of observation $\mathbf{x}_i$. The aforementioned property of the Mann-Whitney U statistic indicates the following:

$$U = P\{R(\mathbf{x_i}|y_i = 1) < R(\mathbf{x_j}|y_j = -1)\}$$

When referring to the rank of an observation, a higher rank is a smaller value, meaning that a rank of 1 is considered higher than a rank of 10, for example. Given this property, it is entirely possible to create pseudo-ROC curves.

ROC curves represent the performance of a binary classifier, based entirely upon how the binary classifier ranks the observations and the true class of these observations. However, if an assumption is made that a classifier has a certain discriminating ability reflected in the AUC or Mann-Whitney U statistic, artificial ranks can be created with pseudo-random numbers. The simplest way to accomplish this is assuming normal distributions for the sake of creating artificial ranks. Suppose A and B are two random variables such that $P(A > B) = U$, or equivalently $P(A - B > 0) = U$. If $W = A - B$, and it is assumed that W is a standard normal random variable, clearly A and B are also normal random variables with a variance of .5. The following table provides examples of this relationship ($(w \sim N(\mu, \sigma^2)$ represents a normal distribution with a mean of μ and a variance of σ^2, with the same notation for the distributions of a and b).

Table 6.1: Examples of Distributions for Creating Artificial Ranks

$P(A > B) = .9$	$w \sim N(1.28, 1)$	$a \sim N(1.28, .5)$	$b \sim N(0, .5)$
$P(A > B) = .8$	$w \sim N(.84, 1)$	$a \sim N(.84, .5)$	$b \sim N(0, .5)$
$P(A > B) = .7$	$w \sim N(.52, 1)$	$a \sim N(.52, .5)$	$b \sim N(0, .5)$

Given the distributions shown in table 6.1, it is now possible to create random numbers which will behave with the desired probability of $P(A > B) = U$. This will also enforce that $P(R(a) > P(R(b)) = U$.

These rankings will now enable the creation of pseudo-ROC curves with an area under the curve equivalent to U. The essence of this method is that it allows for

the study of ROC curves where control variables consist of the AUC, the number of positive examples, and the number of negative examples. An example of five pseudo-ROC curves with $U = .9$ is shown in figure 6.1.

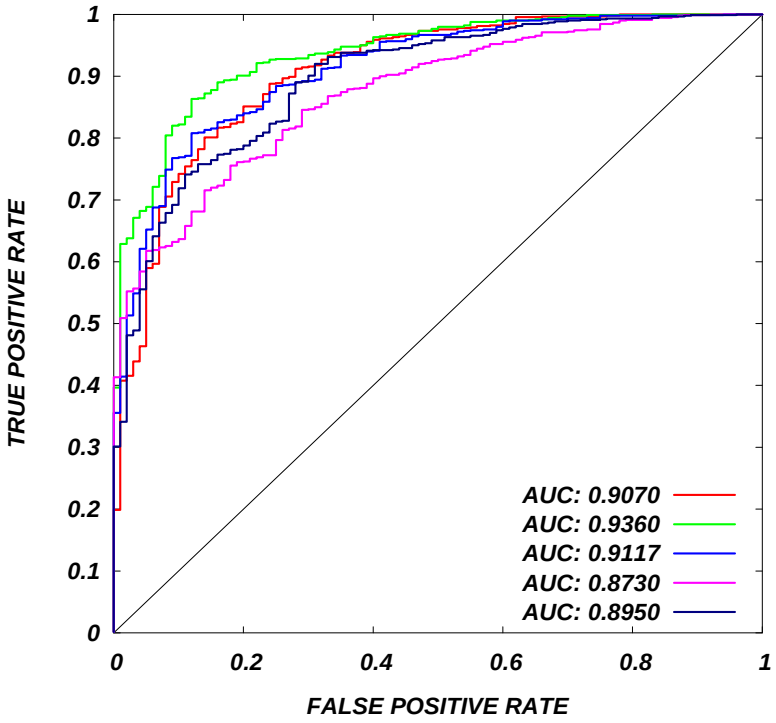

Figure 6.1: Five pseudo-ROC curves

Pseudo-ROC curves serve a multitude of purposes. ROC curves are a very popular method to assess the performance of a binary classifier. Empirical research involving ROC curves has largely been limited to the analysis of curves created by the output of models with real data. The study of ROC curves solely created from the output of classification models limits our ability to fully understand and explore the complete beahvior of ROC curves and ranks. The study of pseudo-ROC curves places a number of parameters into the hands of the researcher - the discriminating power (reflected in the U statistic), proportion of the classes, and total number of observations are all parameters controlled by the researcher with pseudo-ROC curves.

ROC theory research focuses extensively on the topic of the nonparametric statistics which impact ROC curves. This is primarily the Wilcoxon Rank Sum

statistic and Mann-Whitney U statistic [17, 55, 71]. Exceptions to this include [45] and [54, 57] where the authors have taken creative looks at ROC curves to include the application of game theory. However, the concept of the pseudo-ROC curve and use of this method to improve our understanding of ROC curves is a novel approach.

6.4 Rank Distributions

Given a U statistic and desired number of positive and negative instances, it is possible to create rank distributions. Let us consider p positive instances, b negative instances (choosing the letter b to signify a benign or negative observation), and $N = p + b$ total observations. The rank distribution will be a discrete probability distribution, or probability mass function, with $1...N$ possible states, or ranks. Rank distributions reflect the likelihood that a particular rank is a positive or negative observation. For every case there is a given U, p, and b. This information is all that is necessary to create two rank distributions, one for positive observations and one for negative observations.

6.4.1 Utilizing Simulation to Create Rank Distributions

Simulation will be utilized to study these distributions. As stated in [10], estimating probabilities by simulation due to pragmatic necessity (because the analytical solution is very difficult) is an acceptable approach. Given a simulation that models behavior based upon true probabilities, the simulation estimates these probabilities with high accuracy. The combinatoric complexity and implications of order statistics involved with these rank distributions become problematic in creating an analytical solution for the mass functions of the rank distributions. This combinatoric complexity can be shown with a brief example. Suppose that we are interested in a rank distribution with $p = 1$ positive instances and b negative instances. This rank distribution can be solved using the binomial probability distribution. Let r represent the rank of the one positive instance, where $r \in (0, 1, 2, ...b)$. If $r = 0$, the positive instance is ranked first and has come out on top of all negative instances; if $r = b$, the opposite is true (lowest ranking). Given a prediction model with some accuracy, intuition indicates that $P(r = b)$ should be small, and $P(r = 0)$ or at least

the probability that r is close to 0 should be large. This can be modeled as follows:

$$P(r|b) = \binom{b}{r} U^{b-r}(1-U)^r$$

However, this is a trivial case when $p = 1$, which is typically never the case. Given a value of $p > 1$, complexity of the analytical solution grows quickly. There are two ways to attempt to solve the problem analytically for $p > 1$. The first involves attempting to create a discrete probability distribution, similar in essence to the binomial distribution above. This involves managing a distribution that will contain a high degree of combinatoric complexity. The other approach involves studying the order statistics of the underlying distributions that generated the ranks. If q is a continuous random variable which generated the ranks for positive instances, and c is the same for the negative instances, then $(q_{(1)}, q_{(2)}, ..., q_{(p)},)$ and $(c_{(1)}, c_{(2)}, ..., c_{(b)},)$ represent the ordered values of the positive and negative instances, respectively. Again defining $P(r = 0)$ as the probability a positive instance ranks above all other instances, this equates to the following:

$$P(r = 0) = P(q_{(1)} > c_{(1)})$$

This is a relatively simple problem, assuming that the underlying distributions of q and p are known. However, solving the probability that r equates to (1,2,...,b+p) is not as simple.

$$P(r = 1) = P\{(q_{(2)} > c_{(1)}) \cup (c_{(1)} > q_{(1)} > c_{(2)})\}$$

$$P(r = 2) = P\{(q_{(3)} > c_{(1)}) \cup (c_{(1)} > q_{(2)} > c_{(2)}) \cup (c_{(2)} > q_{(1)} > c_{(3)})\}$$

...

The study of these probabilities through simulated rank distributions is much more practical and demonstrates sufficient evidence of the behavior of these rank probabilities.

6.4.2 Behavior of Rank Distributions

These ranking distributions have interesting behavior which directly impact the outcome of ROC curves. First consider two extreme cases. The first extreme case involves a classifier with no predictive power, and in this case $U = .5$. The resulting rank distributions would simply be two uniform distributions ranging between 1 and N (shown as the right plot in Figure 6.2). The other extreme case would be the perfect classifier where $U = 1$. This case would also create two uniform distributions, however the distribution for the ranks of the positive observations would range from $1...p$ and the uniform distribution of the negative observations would range from $p + 1...N$ (shown as the left plot in Figure 6.2).

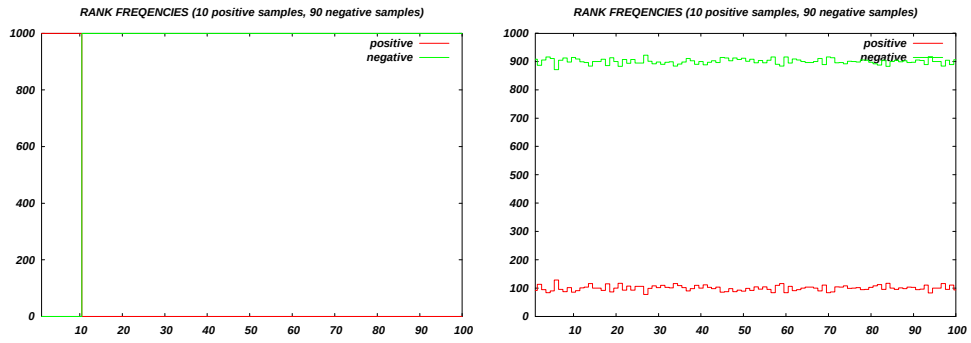

Figure 6.2: The histograms on the left represent perfect classification($U = 1$, and the histograms on the right represent a random (meaningless) classifier ($U = .5$). These histograms resulted from simulations each with 90 positive instances, 10 negative instances, and 1000 simulation runs. The histograms clearly indicate uniform distributions.

Typically, however, U ranges somewhere between .5 and 1. As U increases from .5 to 1, the histogram representing the positive class experiences a reduction in the frequencies of the larger ranks and an increase in the frequencies of the smaller ranks. It is also evident as this shift occurs that the distribution of the positive (minority) class begins to take on the familiar form of the exponential distribution. Figure 6.3 illustrates exactly this phenomenon. This figure illustrates an example which involves 10% positive instances, and each line represents the rank distribution of the positive instances for different values of U. Notice the familiar shape of the exponential distribution emerging as U transitions.

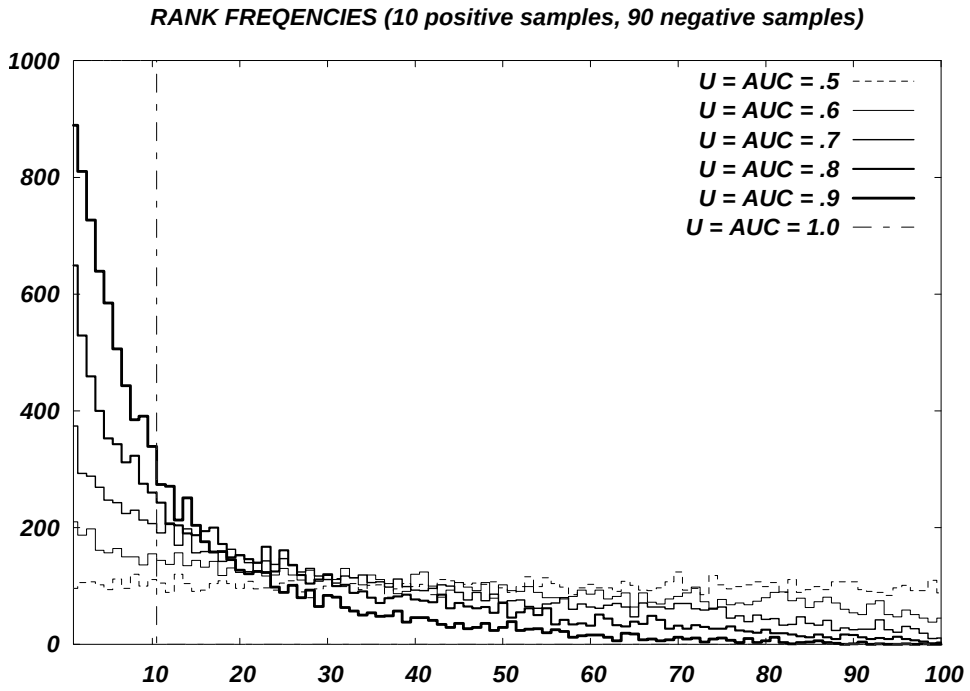

Figure 6.3: The histograms shown above illustrate how the rank frequencies for the minority class transition as U spans the spectrum between .5 and 1.

6.5 Behavior of Fused Classifiers

Analyzing how these rank distributions behave provides insight for model fusion. The fusion metric utilized in the most popular ensemble techniques such as random forest, bagging, and the random subspace method, is the average [18,19,74]. The average is a powerful aggregator, especially if all of the models possess roughly the same predictive power.

Model fusion involves considering several models, all of which measure the same observations, and for each observation fuse the results of each model to arrive at a final decision value for each observation.

Good prediction occurs for a model when the decision value distribution of the positive class achieves separation from the decision value distribution of the negative class. Fusion with the *avg* function invokes the properties of the central limit theorem, and improved separation occurs as a result of variance reduction. This can be further explained in a brief example. Assume that three models each

create a distribution for the decision values of the negative class with a mean of -1 and a variance of 1. Assume the distributions of the positive class have a mean of 1 and a variance of 1. The fused model, using the *avg* aggregator, will create a distribution for the decision values of the negative class with a mean of -1 and a variance of 1/3. The positive class will have a mean of 1 and variance of 1/3. Tighter distributions for both the positive and negative classes creates improved prediction. The fundamental premise of pattern recognition becomes stronger.

6.5.1 Why the Average and min Fusion Metrics Work

The disadvantage of the *avg* aggregator involves the equal weighting and inclusion of all models, good and bad. When fusing security classification problem models, it is likely that some of the models are poor classifiers. Therefore, it is desirable to utilize fusion that is robust against poor classifiers without knowing which classifiers are poor. This is precisely what the min aggregator accomplishes. Given an unbalanced classification problem, the rank distributions which result clearly favor the highest rankings and quickly tail off (see figure 6.3). The behavior is remarkably similar to the exponential distribution. An interesting property of the exponential distribution involves the distribution of the min of this distribution. Given an exponential random variable x distributed with a mean (and standard deviation) of θ, the distribution of the min of x, $x_{(1)}$, is exponential with a mean (and standard deviation) of θ/n. The brief proof of this follows:

$$f(x) = \frac{1}{\theta} e^{-x/\theta}$$

$$F(x) = 1 - e^{-x/\theta}$$

$$F(x_{(1)}) = 1 - (1 - F(x))^n$$

$$F(x_{(1)}) = 1 - e^{-nx/\theta}$$

$$f(x_{(1)}) = \frac{n}{\theta} e^{-nx/\theta}$$

This property indicates that the distribution of $x_{(1)}$ contains less dispersion, concentrating in a tighter range. This concentration enables separation, however

more importantly the min fusion metric creates robustness against poor classifiers. In the security classification problem, we now understand from our study of rank distributions that given a good model the probability of encountering a large rank value for a positive instance is small. It is more likely to observe a small rank value. The min fusion metric indiscriminately eliminates large rank values. This indiscriminant elimination works based on the fact that there are a small number of positive instances. Machine learning researchers will immediately question the contribution of this fusion metric since it is difficult to identify how this fusion metric improves classification. Haykin indicates in [72] that one of the fundamentals in every classification or pattern recognition problems involves ensuring the inclusion of all available information. The min aggregator works based upon our assumption that there is a small number of positive instances. This information contributes and improves performance.

Figure 6.4 illustrates rank distributions for several fusion metrics. This experiment involved fusing a model set where 60% of the models predicted well at a rate of $U = .9$, and 40% of the models create random prediction at a rate of $U = .5$. It is assumed that 10% of the observed instances are positive. Figure 6.4 illustrates the performance of the min, max, and avg fusion metrics. The max fusion metric does not work well. This metric flattens the dispersion of the positive instances, creating poor separation between the positive and negative rank distributions. The avg fusion metric clearly works well, creating two normal distributions produced from the effect of the central limit theorem. These distributions contain minimal overlap. The min aggregator creates the familiar exponential distribution effect, and although there is separation, the quality of the separation is clearly questionable. In the spirit of fuzzy logic, it is possible to combine these metrics. This combination, created simply by computing $\frac{\min + avg}{2}$, creates two distributions which appear tighter than those created from the avg fusion metric without any obvious improvement in performance.

The improvement in performance is not evident until plotting the ROC curves from these rank distributions. Figure 6.5 contains these ROC curves. There are five curves shown, with two models (40%) providing no predictive power ($U \cong .5$). It

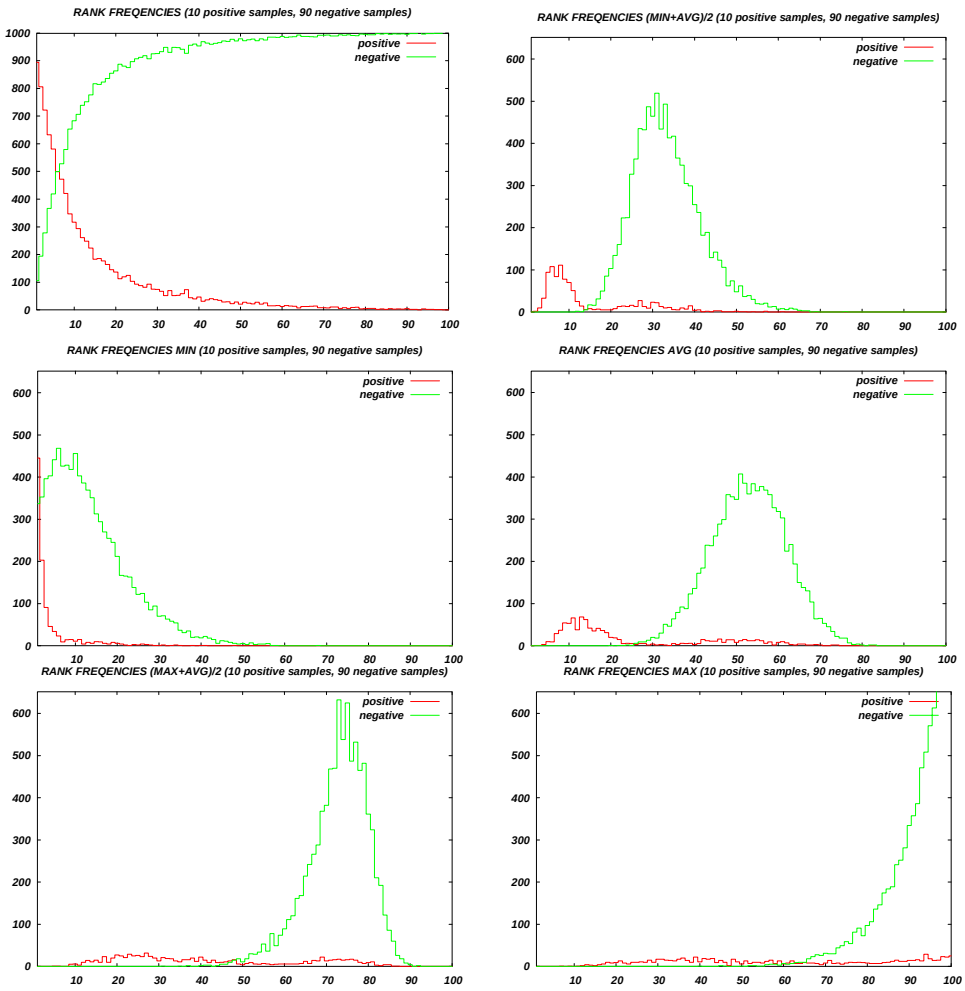

Figure 6.4: The plot in the upper left represents the distributions of the positive and negative classes for a classifier that predicts with a $U = .9$. Simulating that there were 3 classifiers with this accuracy and 2 classifiers with $U = .9$, the other plot represent the performance of various fusion metrics when 60% of the models predict well and 40% predict randomly.

is apparent that the avg and min aggregators perform well, achieving performance close to the convex hull of the ROC curves. However, it is the fusion metric of $\frac{\min + avg}{2}$ which clearly performs best, exceeding the convex hull and creating a synergistic fusion effect.

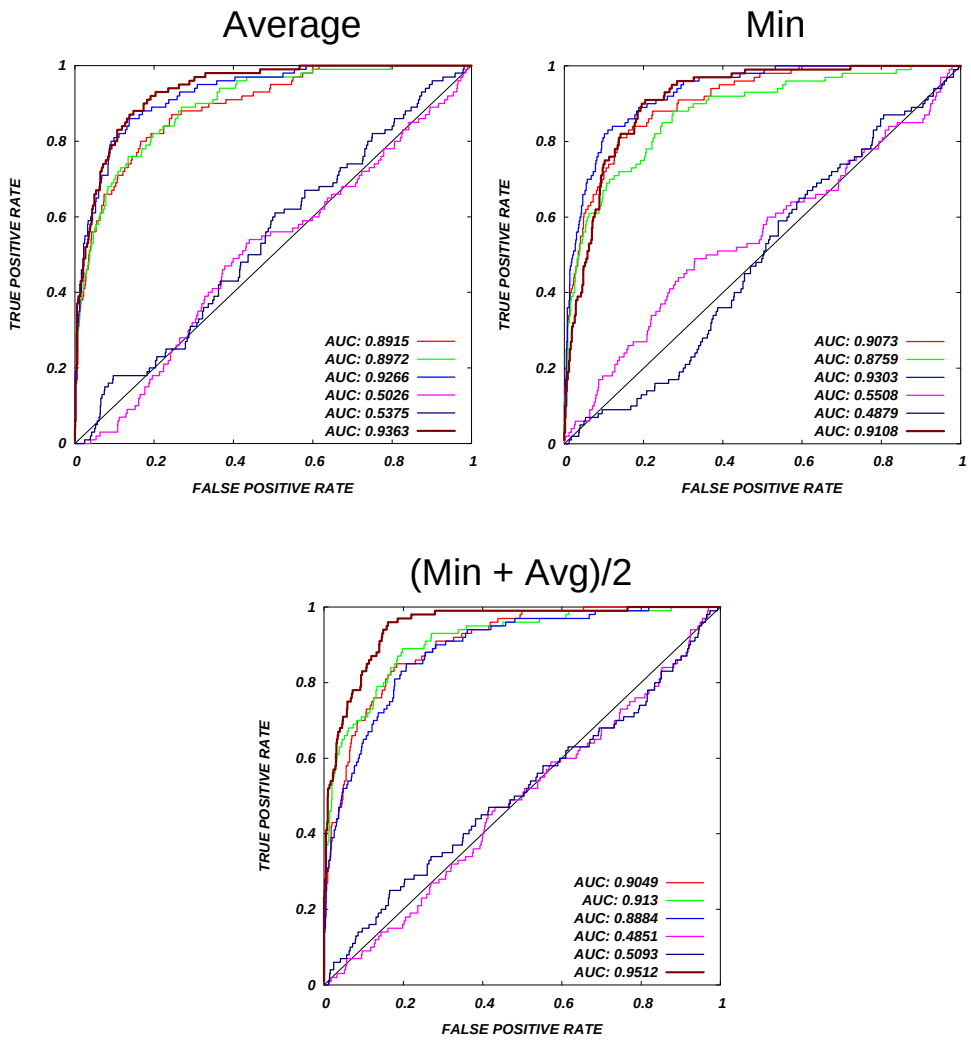

Figure 6.5: The dominant min+ **avg / 2 fusion method was compared against the** min **and avg with a paired t-test, resulting with a** p**-value of nearly 1 illustrating a very high level of statistical significance in the difference.**

6.5.2 An Illustration of Rank Fusion

It is useful to illustrate rank fusion through a toy problem. Table 6.2 shows three models, two of which perform adequately and one which appears to have no predictive power. The resulting ranks created by the min, avg, and min $+avg/2$ functions are also shown. Realize that the resulting columns do not equate to $f(R_1(\mathbf{x}_i), R_2(\mathbf{x}_i), R_3(\mathbf{x}_i)) = f(o_{i1}, o_{i2}, o_{i3})$. The aggregation columns represent $R(f(o_{i1}, o_{i2}, o_{i3}))$. Particularly for the min function, ties must be solved which is

done simply at random. Immediately following any aggregation, decision values are immediately mapped to ranks, or o_{ij}. Table 6.2 is consistent with Algorithm 8 which is detailed later in this chapter; if desired, an interested reader could recreate the last six columns to reinforce the concept.

Table 6.2: A Toy Rank Fusion Problem

True Class	$\mathbf{R}_1$	$\mathbf{R}_2$	$\mathbf{R}_3$	$\mathbf{R}_4$	$\mathbf{R}_5$	min	avg	R(min) $= o_{i2}$	R(avg) $= o_{i2}$	$\frac{\text{min}+\text{avg}}{2}$ $= \frac{o_{i2}+o_{i2}}{2}$	$R(\frac{\text{min}+\text{avg}}{2})$
0	8	15	5	6	10	5.005	8.807	16	8	12.005	12
0	16	16	19	7	17	7.006	15.003	18	20	19.000	19
0	17	18	13	15	9	9.008	14.400	19	17	18.004	18
0	15	20	8	19	1	1.002	12.601	2	15	8.504	8
0	11	13	7	1	2	1.008	6.804	4	3	3.509	3
0	14	9	4	9	18	4.007	10.808	12	12	12.008	13
0	6	6	9	18	3	3.004	8.402	10	5	7.503	6
0	13	7	20	10	5	5.003	11.007	15	13	14.008	16
0	18	17	11	3	13	3.002	12.410	8	14	11.006	11
0	7	8	16	4	7	4.003	8.402	11	4	7.508	7
0	19	19	6	8	19	6.001	14.202	17	16	16.510	17
0	12	11	14	20	15	11.003	14.404	20	18	19.008	20
0	4	10	15	5	11	4.009	9.000	13	9	11.005	10
0	5	12	17	11	6	5.003	10.209	14	11	12.509	14
0	20	3	18	17	16	3.001	14.801	7	19	13.007	15
1	3	2	10	16	12	2.001	8.605	5	6	5.508	5
1	9	4	3	14	20	3.003	10.007	9	10	9.509	9
1	1	5	1	12	14	1.003	6.609	3	2	2.508	1
1	10	1	12	13	8	1.001	8.804	1	7	4.005	4
1	2	14	2	2	4	2.005	4.801	6	1	3.505	2
AUC	0.87	0.85	0.83	0.44	0.43	-	-	0.88	0.853	-	0.920

6.6 The Properties of Synergistic Fusion - a Factorial Design Illustration

There is a fundamental synergistic fusion property discussed in this chapter. Stated simply, this property claims that when fusing ranks, there is improved discriminating power from the min aggregator if the problem is unbalanced with a minority positive class. The opposite is true for the max aggregator if the problem

is unbalanced with a minority negative class. The chapter supports this property with a discussion that includes a statistical explanation of the behavior of ranks created in a classification problem as well as the scenario depicted in figure 6.5. Figure 6.5 provides results for only one scenario, or parameter set. This begs a question: how does this fusion strategy behave across the spectrum of each parameter? A way to address this question involves creating a factorial design. Factorial design stems from a body of knowledge known as design of experiments, or DOE. R.A. Fisher was a pioneer of DOE, and researchers give much credit to Fisher for the current studies involving DOE [16]. For a comprehensive collection of Fisher's work to include his DOE work, see [59].

The DOE for this problem involved four factors. These factors were derived from simply considering what parameters effect this fusion problem. These parameters, or factors, include:

- **balance**: the number of observations (out of 1000) which are members of the negative class. This factor becomes the most important factor in the experiment. The primary hypothesis of this study claims that the balance of the problem closely relates to the utility of the min aggregator. This experiment will reinforce this hypothesis.

- **AUC of "good" models**: the assumed area under the curve (AUC) or predictive power of effective models. A major assumption includes that all of the "good" models predict with a specific accuracy.

- **fraction "good"**: the percent of models predicting with the accuracy of the AUC, and all others have no predictive power (AUC = .5). If all of the models are "good", the average (avg) aggregator suffices. The min aggregator is more robust against these powerless classifiers. This is simply by favoring smaller ranks which statistically tend to be members of the positive class for good models classifying in an imbalanced environment (positive minority). The avg aggregator considers all of the models equally, and therefore becomes susceptible to meaningless ranks created by the "poor" models.

Table 6.3: Design of Experiments

	balance	AUC of good models	fraction good	number of models
min value	500	0.8	0.4	2
max value	980	0.95	1	8
step	160	0.05	0.2	2

- **number of models**: the number of models fused. This has an important but subtle effect on the outcome that will be discussed later in the chapter.

There were 4 levels explored for each factor creating $4^4 = 256$ design points (dp). Each design point consisted of 30 repetitions, each repetition utilizing 1000 observations for each experiment. Three different fusion strategies were employed at each design point: min, $(\min + avg)/2$, and avg.

Table 6.3 illustrates the values of the parameters considered in the DOE. The balance of the class ranges from 500 negative instances (completely balanced) to 980 negative instances (severely imbalanced) with a step of 160 between levels. The 'balance' parameter is not explored for a minority negative class. This is because the exact same property observed with the min aggregator for the minority positive class can be observed for the max aggregator if a minority negative class is considered. It is a symmetrical property that will not be shown for the sake of limiting redundancy. The AUC of the "good" models ranges from .8 to .95 with a step of .05 between levels. There was definitely a bias to explore the higher end of AUC values; further experimentation should consider exploring the lower range of AUC values. The fraction of "good" models replicates the unknown sensors or models in the ensemble which predict well, assuming that the others predict randomly (AUC = .5). This fraction ranges between .4 and 1 with a .2 step, exploring a slight minority of "good" models to observing a majority of "good" models. Brief experimentation indicated that having a fraction of "good" models less than .4 created poor and inconsistent performance across the board. The number of models ranged from 2 to 8 with a step of 2. Larger numbers of models severely favor the avg aggregator. Central

limit theorem becomes stronger as the number of models increases therefore favoring the *avg* aggregator. The min aggregator works based upon the simple premise that positive instances are more likely to rank as a low number when considering imbalanced (positive minority) problems. Given an imbalanced problem (minority positive class) with a small number of models to fuse, the min aggregator is less likely than the *avg* aggregator to be affected by a random model.

Tables 6.4, 6.5, and 6.6 illustrate the results of the DOE. Every experimental result is shown in this table for the sake of complete transparency of this experiment. An interested reader may want to inspect various design points, and all are present to inspect. The sorted order of the design points is intentional. The design points are ordered from the balanced case (500 negative instance) to the imbalanced case (980 negative instances). Inspection of the tables will reveal that the min aggregator performs better as imbalance increases. The $(\min + avg)/2$ aggregator continues to present the best performance for higher imbalance. This is a heuristic, no doubt, but it is a heuristic that is supported with theoretical discussion and close ties to fuzzy logic. The discriminating power of the central limit theorem surfaces from the *avg* aggregator, and the min aggregator provides robustness against random models for imbalanced scenarios.

Figure 6.6 compares the performance of the avg aggregator and the $(\min + avg)/2$ aggregator. The min aggregator is not plotted or discussed as it is inferior as a stand alone aggregator. The balance factor is the critical factor. Scatter plots exploring the spectrum of the other factors is entirely possible and has been explored, however these factors have no effect, plotting a scatter plot which essentially straddles the diagonal.

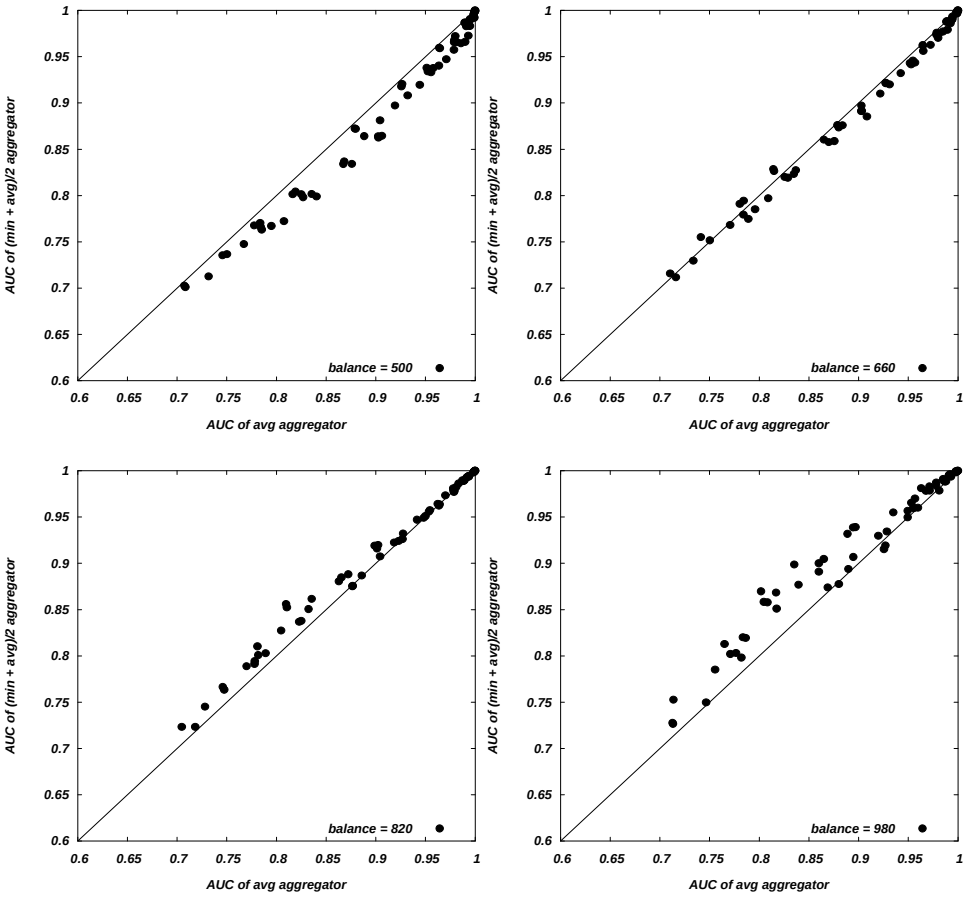

Figure 6.6: These plots compare the AUC achieved when fusing with the avg aggregator versus the $(\min + avg)/2$ aggregator. Every plot contains the full spectrum of design points with the exception of the balance factor which is held constant for each plot. Notice that as imbalance increases, the $(\min + avg)/2$ aggregator becomes more dominant.

Several test statistics can serve to illuminate this experiment. One of the most interesting statistical tests is an analysis of variance (ANOVA) for the four

Table 6.4: Experimental results and full factorial table - part 1.

	balance	AUC of good models	fraction good	number of models	min AUC	min+avg AUC	avg AUC
dp1	500	0.8	0.4	2	0.6787	0.701	0.7084
dp2	500	0.85	0.4	2	0.7121	0.7366	0.7502
dp3	500	0.9	0.4	2	0.7376	0.7703	0.7837
dp4	500	0.95	0.4	2	0.766	0.8042	0.8192
dp5	500	0.8	0.6	2	0.6816	0.7025	0.7074
dp6	500	0.85	0.6	2	0.708	0.7354	0.7458
dp7	500	0.9	0.6	2	0.7414	0.7677	0.7777
dp8	500	0.95	0.6	2	0.764	0.8014	0.8164
dp9	500	0.8	0.8	2	0.848	0.8725	0.8789
dp10	500	0.85	0.8	2	0.8972	0.9178	0.9258
dp11	500	0.9	0.8	2	0.9406	0.9596	0.9639
dp12	500	0.95	0.8	2	0.9771	0.9871	0.9897
dp13	500	0.8	1	2	0.8458	0.872	0.8797
dp14	500	0.85	1	2	0.8965	0.9206	0.9265
dp15	500	0.9	1	2	0.9413	0.9591	0.9646
dp16	500	0.95	1	2	0.9768	0.9865	0.9894
dp17	500	0.8	0.4	4	0.6981	0.7661	0.7841
dp18	500	0.85	0.4	4	0.7256	0.8013	0.825
dp19	500	0.9	0.4	4	0.7463	0.8368	0.8683
dp20	500	0.95	0.4	4	0.7607	0.8639	0.9025
dp21	500	0.8	0.6	4	0.6971	0.7631	0.7853
dp22	500	0.85	0.6	4	0.7303	0.7981	0.8269
dp23	500	0.9	0.6	4	0.7471	0.834	0.8674
dp24	500	0.95	0.6	4	0.7582	0.8625	0.9027
dp25	500	0.8	0.8	4	0.7938	0.8641	0.8884
dp26	500	0.85	0.8	4	0.8271	0.9081	0.9322
dp27	500	0.9	0.8	4	0.8592	0.9403	0.9636
dp28	500	0.95	0.8	4	0.8775	0.9644	0.9858
dp29	500	0.8	1	4	0.8888	0.9379	0.9514
dp30	500	0.85	1	4	0.9324	0.9722	0.98
dp31	500	0.9	1	4	0.9669	0.9906	0.9949
dp32	500	0.95	1	4	0.9896	0.9986	0.9995
dp33	500	0.8	0.4	6	0.6402	0.7126	0.7318
dp34	500	0.85	0.4	6	0.6553	0.7475	0.7673
dp35	500	0.9	0.4	6	0.6699	0.7722	0.8076
dp36	500	0.95	0.4	6	0.6795	0.7989	0.8404
dp37	500	0.8	0.6	6	0.7719	0.8813	0.9043
dp38	500	0.85	0.6	6	0.8022	0.9196	0.9444
dp39	500	0.9	0.6	6	0.8261	0.9472	0.971
dp40	500	0.95	0.6	6	0.8361	0.966	0.99
dp41	500	0.8	0.8	6	0.8387	0.9341	0.9524
dp42	500	0.85	0.8	6	0.8796	0.9657	0.9788
dp43	500	0.9	0.8	6	0.9036	0.9834	0.9941
dp44	500	0.95	0.8	6	0.9173	0.9926	0.999
dp45	500	0.8	1	6	0.9042	0.9679	0.9791
dp46	500	0.85	1	6	0.9468	0.9889	0.9945
dp47	500	0.9	1	6	0.9767	0.9976	0.9991
dp48	500	0.95	1	6	0.9942	0.9997	1
dp49	500	0.8	0.4	8	0.6611	0.767	0.795
dp50	500	0.85	0.4	8	0.674	0.8016	0.8355
dp51	500	0.9	0.4	8	0.6897	0.8341	0.876
dp52	500	0.95	0.4	8	0.6945	0.8644	0.9063
dp53	500	0.8	0.6	8	0.7642	0.8973	0.9194
dp54	500	0.85	0.6	8	0.7959	0.9331	0.9557
dp55	500	0.9	0.6	8	0.8095	0.9574	0.9787
dp56	500	0.95	0.6	8	0.8191	0.9727	0.9932
dp57	500	0.8	0.8	8	0.8188	0.9378	0.9579
dp58	500	0.85	0.8	8	0.8461	0.9656	0.9817
dp59	500	0.9	0.8	8	0.8646	0.9831	0.9949
dp60	500	0.95	0.8	8	0.8755	0.9919	0.9991
dp61	500	0.8	1	8	0.9167	0.9829	0.991
dp62	500	0.85	1	8	0.9548	0.9953	0.9982
dp63	500	0.9	1	8	0.9815	0.9992	0.9999
dp64	500	0.95	1	8	0.9957	1	1
dp65	660	0.8	0.4	2	0.6965	0.7116	0.7162
dp66	660	0.85	0.4	2	0.7323	0.7515	0.7503
dp67	660	0.9	0.4	2	0.7755	0.7943	0.7843
dp68	660	0.95	0.4	2	0.8093	0.8263	0.815
dp69	660	0.8	0.6	2	0.6975	0.7156	0.7105
dp70	660	0.85	0.6	2	0.7345	0.755	0.7413
dp71	660	0.9	0.6	2	0.7754	0.7909	0.7805
dp72	660	0.95	0.6	2	0.8103	0.8285	0.8143
dp73	660	0.8	0.8	2	0.8446	0.8734	0.8801
dp74	660	0.85	0.8	2	0.8941	0.9214	0.9274
dp75	660	0.9	0.8	2	0.9429	0.9621	0.9648
dp76	660	0.95	0.8	2	0.9771	0.9878	0.9883
dp77	660	0.8	1	2	0.8459	0.876	0.8788
dp78	660	0.85	1	2	0.8955	0.9218	0.9274
dp79	660	0.9	1	2	0.9407	0.9624	0.9646
dp80	660	0.95	1	2	0.9766	0.9885	0.989
dp81	660	0.8	0.4	4	0.7208	0.7746	0.7891
dp82	660	0.85	0.4	4	0.7568	0.8192	0.8289
dp83	660	0.9	0.4	4	0.786	0.8603	0.8651
dp84	660	0.95	0.4	4	0.8123	0.8914	0.9032
dp85	660	0.8	0.6	4	0.7225	0.7792	0.7841

Table 6.5: Experimental results and full factorial table - part 2.

	balance	AUC of good models	fraction good	number of models	min AUC	min+avg AUC	avg AUC
dp86	660	0.85	0.6	4	0.7585	0.82	0.8256
dp87	660	0.9	0.6	4	0.7878	0.8576	0.8701
dp88	660	0.95	0.6	4	0.8085	0.8912	0.9027
dp89	660	0.8	0.8	4	0.808	0.8758	0.8839
dp90	660	0.85	0.8	4	0.8517	0.92	0.9312
dp91	660	0.9	0.8	4	0.8866	0.9563	0.9651
dp92	660	0.95	0.8	4	0.9101	0.9772	0.985
dp93	660	0.8	1	4	0.8866	0.943	0.9518
dp94	660	0.85	1	4	0.9321	0.9758	0.9786
dp95	660	0.9	1	4	0.9673	0.9932	0.9945
dp96	660	0.95	1	4	0.9902	0.9991	0.9994
dp97	660	0.8	0.4	6	0.6674	0.7295	0.7337
dp98	660	0.85	0.4	6	0.6909	0.7681	0.7708
dp99	660	0.9	0.4	6	0.7122	0.797	0.8091
dp100	660	0.95	0.4	6	0.7271	0.8274	0.8369
dp101	660	0.8	0.6	6	0.7932	0.8971	0.9029
dp102	660	0.85	0.6	6	0.8285	0.9322	0.9424
dp103	660	0.9	0.6	6	0.8642	0.9628	0.9725
dp104	660	0.95	0.6	6	0.8769	0.9791	0.9896
dp105	660	0.8	0.8	6	0.8537	0.9417	0.953
dp106	660	0.85	0.8	6	0.8925	0.9739	0.9788
dp107	660	0.9	0.8	6	0.9217	0.9901	0.994
dp108	660	0.95	0.8	6	0.94	0.9969	0.9991
dp109	660	0.8	1	6	0.9041	0.9738	0.9785
dp110	660	0.85	1	6	0.9473	0.9916	0.9944
dp111	660	0.9	1	6	0.9764	0.9982	0.9992
dp112	660	0.95	1	6	0.9939	0.9999	1
dp113	660	0.8	0.4	8	0.6852	0.7851	0.7959
dp114	660	0.85	0.4	8	0.7134	0.8232	0.8349
dp115	660	0.9	0.4	8	0.7353	0.8588	0.8756
dp116	660	0.95	0.4	8	0.7479	0.8853	0.9084
dp117	660	0.8	0.6	8	0.7909	0.91	0.9219
dp118	660	0.85	0.6	8	0.8235	0.9436	0.9568
dp119	660	0.9	0.6	8	0.8495	0.9704	0.9801
dp120	660	0.95	0.6	8	0.8609	0.986	0.9926
dp121	660	0.8	0.8	8	0.8357	0.9458	0.9548
dp122	660	0.85	0.8	8	0.8687	0.9752	0.981
dp123	660	0.9	0.8	8	0.8965	0.9901	0.994
dp124	660	0.95	0.8	8	0.9117	0.9967	0.9991
dp125	660	0.8	1	8	0.9141	0.9871	0.9901
dp126	660	0.85	1	8	0.956	0.9967	0.9979
dp127	660	0.9	1	8	0.9812	0.9996	0.9999
dp128	660	0.95	1	8	0.9958	1	1
dp129	820	0.8	0.4	2	0.7165	0.7232	0.7048
dp130	820	0.85	0.4	2	0.7669	0.7634	0.7475
dp131	820	0.9	0.4	2	0.8102	0.8102	0.781
dp132	820	0.95	0.4	2	0.8562	0.8524	0.8105
dp133	820	0.8	0.6	2	0.7195	0.7233	0.7183
dp134	820	0.85	0.6	2	0.7666	0.7664	0.746
dp135	820	0.9	0.6	2	0.8104	0.8008	0.7817
dp136	820	0.95	0.6	2	0.8574	0.856	0.8098
dp137	820	0.8	0.8	2	0.8484	0.8756	0.8768
dp138	820	0.85	0.8	2	0.8965	0.9242	0.9231
dp139	820	0.9	0.8	2	0.9415	0.9623	0.9637
dp140	820	0.95	0.8	2	0.9769	0.9893	0.9885
dp141	820	0.8	1	2	0.8431	0.8751	0.8766
dp142	820	0.85	1	2	0.8928	0.9262	0.9269
dp143	820	0.9	1	2	0.9424	0.9637	0.9643
dp144	820	0.95	1	2	0.9762	0.9889	0.9885
dp145	820	0.8	0.4	4	0.7479	0.7912	0.778
dp146	820	0.85	0.4	4	0.7948	0.8378	0.825
dp147	820	0.9	0.4	4	0.8384	0.8804	0.8629
dp148	820	0.95	0.4	4	0.8725	0.919	0.8989
dp149	820	0.8	0.6	4	0.7482	0.7942	0.7782
dp150	820	0.85	0.6	4	0.7944	0.8368	0.823
dp151	820	0.9	0.6	4	0.8358	0.8847	0.8655
dp152	820	0.95	0.6	4	0.8742	0.9197	0.9024
dp153	820	0.8	0.8	4	0.8214	0.8867	0.8859
dp154	820	0.85	0.8	4	0.8736	0.9321	0.9275
dp155	820	0.9	0.8	4	0.9118	0.9642	0.9624
dp156	820	0.95	0.8	4	0.9415	0.9863	0.9835
dp157	820	0.8	1	4	0.8856	0.9492	0.9483
dp158	820	0.85	1	4	0.9306	0.9785	0.9791
dp159	820	0.9	1	4	0.9661	0.9941	0.9934
dp160	820	0.95	1	4	0.9902	0.9993	0.999
dp161	820	0.8	0.4	6	0.7031	0.745	0.7282
dp162	820	0.85	0.4	6	0.7371	0.7888	0.77
dp163	820	0.9	0.4	6	0.7761	0.8274	0.8048
dp164	820	0.95	0.4	6	0.8024	0.8615	0.8355
dp165	820	0.8	0.6	6	0.8216	0.9075	0.9044
dp166	820	0.85	0.6	6	0.8674	0.9471	0.9415
dp167	820	0.9	0.6	6	0.8993	0.9735	0.9701
dp168	820	0.95	0.6	6	0.9241	0.9897	0.9873
dp169	820	0.8	0.8	6	0.8664	0.951	0.95
dp170	820	0.85	0.8	6	0.9093	0.979	0.9781

Table 6.6: Experimental results and full factorial table - part 3.

	balance	AUC of good models	fraction good	number of models	min AUC	min+avg AUC	avg AUC
dp171	820	0.9	0.8	6	0.9432	0.9935	0.9926
dp172	820	0.95	0.8	6	0.9631	0.9989	0.9987
dp173	820	0.8	1	6	0.9055	0.9771	0.9787
dp174	820	0.85	1	6	0.944	0.9935	0.9937
dp175	820	0.9	1	6	0.9765	0.9988	0.9989
dp176	820	0.95	1	6	0.9944	1	1
dp177	820	0.8	0.4	8	0.7262	0.8029	0.7892
dp178	820	0.85	0.4	8	0.7668	0.8505	0.8324
dp179	820	0.9	0.4	8	0.8024	0.8881	0.8724
dp180	820	0.95	0.4	8	0.8264	0.9159	0.9013
dp181	820	0.8	0.6	8	0.8251	0.9225	0.9186
dp182	820	0.85	0.6	8	0.8631	0.9575	0.9545
dp183	820	0.9	0.6	8	0.8948	0.9809	0.9779
dp184	820	0.95	0.6	8	0.9155	0.9927	0.9915
dp185	820	0.8	0.8	8	0.8559	0.9557	0.9535
dp186	820	0.85	0.8	8	0.9015	0.9825	0.981
dp187	820	0.9	0.8	8	0.9287	0.9946	0.9938
dp188	820	0.95	0.8	8	0.9468	0.9988	0.999
dp189	820	0.8	1	8	0.915	0.99	0.9893
dp190	820	0.85	1	8	0.9562	0.998	0.998
dp191	820	0.9	1	8	0.9812	0.9998	0.9998
dp192	820	0.95	1	8	0.9954	1	1
dp193	980	0.8	0.4	2	0.728	0.7265	0.7133
dp194	980	0.85	0.4	2	0.7938	0.7497	0.7466
dp195	980	0.9	0.4	2	0.857	0.82	0.7837
dp196	980	0.95	0.4	2	0.9124	0.8583	0.8047
dp197	980	0.8	0.6	2	0.7288	0.7275	0.7128
dp198	980	0.85	0.6	2	0.7895	0.7851	0.7556
dp199	980	0.9	0.6	2	0.8462	0.8127	0.7651
dp200	980	0.95	0.6	2	0.9169	0.8697	0.8019
dp201	980	0.8	0.8	2	0.8525	0.874	0.869
dp202	980	0.85	0.8	2	0.8927	0.9153	0.9256
dp203	980	0.9	0.8	2	0.9473	0.96	0.9598
dp204	980	0.95	0.8	2	0.9729	0.9893	0.986
dp205	980	0.8	1	2	0.8451	0.8775	0.8802
dp206	980	0.85	1	2	0.8946	0.9193	0.9271
dp207	980	0.9	1	2	0.9383	0.9595	0.9558
dp208	980	0.95	1	2	0.9734	0.9883	0.9868
dp209	980	0.8	0.4	4	0.7815	0.802	0.771
dp210	980	0.85	0.4	4	0.8349	0.851	0.8175
dp211	980	0.9	0.4	4	0.895	0.8909	0.8601
dp212	980	0.95	0.4	4	0.9538	0.9391	0.8969
dp213	980	0.8	0.6	4	0.7745	0.7981	0.7821
dp214	980	0.85	0.6	4	0.8396	0.8684	0.817
dp215	980	0.9	0.6	4	0.8981	0.9	0.8601
dp216	980	0.95	0.6	4	0.9499	0.9318	0.8889
dp217	980	0.8	0.8	4	0.8529	0.8939	0.8898
dp218	980	0.85	0.8	4	0.9007	0.9343	0.9285
dp219	980	0.9	0.8	4	0.9479	0.9699	0.9568
dp220	980	0.95	0.8	4	0.9764	0.9873	0.9782
dp221	980	0.8	1	4	0.89	0.9497	0.9494
dp222	980	0.85	1	4	0.928	0.9786	0.9812
dp223	980	0.9	1	4	0.9637	0.9936	0.993
dp224	980	0.95	1	4	0.9916	0.9992	0.9988
dp225	980	0.8	0.4	6	0.7416	0.7525	0.7137
dp226	980	0.85	0.4	6	0.8015	0.803	0.7767
dp227	980	0.9	0.4	6	0.8753	0.8578	0.8084
dp228	980	0.95	0.4	6	0.921	0.8986	0.8353
dp229	980	0.8	0.6	6	0.8559	0.9068	0.8947
dp230	980	0.85	0.6	6	0.9109	0.9549	0.935
dp231	980	0.9	0.6	6	0.9474	0.9812	0.9631
dp232	980	0.95	0.6	6	0.9788	0.9912	0.9853
dp233	980	0.8	0.8	6	0.8823	0.9567	0.9494
dp234	980	0.85	0.8	6	0.9268	0.9786	0.9716
dp235	980	0.9	0.8	6	0.9669	0.9946	0.9928
dp236	980	0.95	0.8	6	0.9862	0.9993	0.9986
dp237	980	0.8	1	6	0.9059	0.9782	0.9678
dp238	980	0.85	1	6	0.9453	0.9951	0.9911
dp239	980	0.9	1	6	0.9762	0.9991	0.9981
dp240	980	0.95	1	6	0.9935	0.9999	0.9997
dp241	980	0.8	0.4	8	0.7723	0.8194	0.7865
dp242	980	0.85	0.4	8	0.8461	0.8769	0.8396
dp243	980	0.9	0.4	8	0.9014	0.9046	0.865
dp244	980	0.95	0.4	8	0.9515	0.9388	0.8945
dp245	980	0.8	0.6	8	0.8556	0.9297	0.92
dp246	980	0.85	0.6	8	0.9076	0.96	0.9552
dp247	980	0.9	0.6	8	0.9547	0.983	0.9714
dp248	980	0.95	0.6	8	0.9785	0.9952	0.9911
dp249	980	0.8	0.8	8	0.8872	0.9654	0.9532
dp250	980	0.85	0.8	8	0.9254	0.9837	0.9781
dp251	980	0.9	0.8	8	0.9677	0.9963	0.992
dp252	980	0.95	0.8	8	0.9879	0.9993	0.9978
dp253	980	0.8	1	8	0.921	0.9886	0.9879
dp254	980	0.85	1	8	0.9542	0.9981	0.9979
dp255	980	0.9	1	8	0.9834	0.9998	0.9996
dp256	980	0.95	1	8	0.9958	1	1

factor experiment. The primary reference for this analysis was [118]. Multi-factor ANOVA is a compact method to analyze the effect of each individual factors as well as the interaction between factors. As factors in an experiment grow, complexity grows quickly as well. Furthermore, a large number of factors creates challenges in explaining multifactor interaction with many factors [118]. The statistic, or yield as it often is called in DOE, analyzed in table 6.7 is the difference between the avg aggregator and the (min + avg)/2 aggregator. The purpose of the analysis is to determine whether or not additional evidence exists supporting the hypothesis that the (min + avg)/2 aggregator works well. It is necessary to define a few terms and statistics used in this experiment. Detailed discussion of the statistics can be found in [118]. The long list of formulas has been included below to show the complexity and potential for explosive growth if one wanted to analyze more factors with this type of DOE analysis. Using the conventions and methods shown by Peterson in [118], the following terms support the ANOVA summary for this experiment:

$y_{ijkl} = \frac{\min + \text{avg}}{2} - \text{avg}$

The subscripts represent the i^{th} level of 'balance' (factor A), j^{th} level of 'AUC of good models' (factor B), k^{th} level of 'fraction good' (factor C), l^{th} level of 'number of models' (factor D). The $\frac{\min + \text{avg}}{2}$ and avg are the average result of thirty repetitions conducted for that design point. Furthermore, $(i, j, k, l) \in (1, 2, 3, 4)$.

$G = \sum_i \sum_j \sum_k \sum_l y_{ijkl}$

$C = \frac{G^2}{rabc}$

$\text{SSTot} = \sum_i \sum_j \sum_k \sum_l y_{ijkl}^2 - C$

$\text{SSA} = \sum_i (1/64)(\sum_j \sum_k \sum_l y_{ijkl})^2 - C$

$\text{SSB} = \sum_j (1/64)(\sum_i \sum_k \sum_l y_{ijkl})^2 - C$

$\text{SSC} = \sum_k (1/64)(\sum_i \sum_j \sum_l y_{ijkl})^2 - C$

$\text{SSD} = \sum_l (1/64)(\sum_i \sum_j \sum_k y_{ijkl})^2 - C$

$\text{SSAB} = \sum_i \sum_j (1/16)(\sum_k \sum_l y_{ijkl}^2) - C - \text{SSA} - \text{SSB}$

$\text{SSAC} = \sum_i \sum_k (1/16)(\sum_j \sum_l y_{ijkl}^2) - C - \text{SSA} - \text{SSC}$

$\text{SSAD} = \sum_i \sum_l (1/16)(\sum_j \sum_k y_{ijkl}^2) - C - \text{SSA} - \text{SSD}$

$\text{SSBC} = \sum_j \sum_k (1/16)(\sum_i \sum_l y_{ijkl}^2) - C - \text{SSA} - \text{SSD}$

$\text{SSBD} = \sum_j \sum_l (1/16)(\sum_i \sum_k y_{ijkl}^2) - C - \text{SSB} - \text{SSD}$

$$\text{SSCD} = \sum_k \sum_l (1/16)(\sum_i \sum_j y_{ijkl}^2) - C - \text{SSC} - \text{SSD}$$

$$\text{SSABC} = \sum_i \sum_j \sum_k (1/4)(\sum_l y_{ijkl}^2) - C - \text{SSA} - \text{SSB} - \text{SSC} - \text{SSAB} - \text{SSBC}$$

$$\text{SSABD} = \sum_i \sum_j \sum_l (1/4)(\sum_k y_{ijkl}^2) - C - \text{SSA} - \text{SSB} - \text{SSD} - \text{SSAB} - \text{SSBD}$$

$$\text{SSACD} = \sum_i \sum_k \sum_l (1/4)(\sum_j y_{ijkl}^2) - C - \text{SSA} - \text{SSC} - \text{SSD} - \text{SSAC} - \text{SSCD}$$

$$\text{SSBCD} = \sum_j \sum_k \sum_l (1/4)(\sum_i y_{ijkl}^2) - C - \text{SSB} - \text{SSC} - \text{SSD} - \text{SSBC} - \text{SSCD}$$

$$\text{SSABCD} = \sum_i \sum_j \sum_k \sum_l y_{ijkl}^2 - C - \text{SSA} - \text{SSB} - \text{SSC} - \text{SSD} - \text{SSAB} - \text{SSAC} - \text{SSAD} - \text{SSBC} - \text{SSBD} - \text{SSCD} - \text{SSABC} - \text{SSABD} - \text{SSACD} - -\text{SSBCD}$$

$$\text{SSE} = \text{SSTot} - \text{SSA} - \text{SSB} - \text{SSC} - \text{SSD} - \text{SSAB} - \text{SSAC} - \text{SSAD} - \text{SSBC} - \text{SSBD} - \text{SSCD} - \text{SSABC} - \text{SSABD} - \text{SSACD} - -\text{SSBCD}$$

Table 6.7: ANOVA summary table for factorial experiment.

	d.f.	SS	MS	F	p value
A	3	406.04	135.35	223.10	1.00
B	3	5.36	1.79	2.95	0.95
C	3	32.27	10.76	17.73	0.99
D	3	21.36	7.12	11.74	0.99
AB	9	6.37	0.71	1.17	0.65
AC	9	189.97	21.11	34.79	0.99
AD	9	15.18	1.69	2.78	0.98
BC	9	1.27	0.14	0.23	0.01
BD	9	10.78	1.20	1.97	0.91
CD	9	35.78	3.98	6.55	0.99
ABC	27	28.19	1.04	1.72	0.91
ABD	27	10.57	0.39	0.65	0.13
ACD	27	33.58	1.24	2.05	0.96
BCD	27	10.49	0.39	0.64	0.13
error	26	15.77	0.61		

The MS can be calculated by dividing the SS by the degrees of freedom (d.f.). The F statistic is the mean square statistic divided by the MSE, which is .61 in our experiment. The 'balance' factor, shown first and represented as A in the table, clearly has a strong influence on the yield in our model (recall that we defined the yield as the difference between (min + avg)/2 and avg. This strong relationship is expected based on the plots shown in figure 6.6. It is also interesting to note that the 'fraction good' and 'number of models' have a strong impact on the yield. This makes

sense. As the number of models increases, the average becomes a much stronger aggregator. If more than eight models are to be fused, the average will perform the best. This relationship with the yield is evident in the correlations shown in table 6.8. Some of the strong interactions revealed in the experiment deserve discussion as well. The ANOVA table shows where interactions exist, however coupled with the correlations the direction of the relationship becomes more apparent. If the fraction of good models is large, the average aggregator again will dominate. The purpose of the min aggregator is to provide robustness against week or completely useless classifiers which the decision make does not have the ability to identify or eliminate. It is interesting to note that the AUC of the good models seems to have a minimal effect. Part of this reason could be that the span of the AUCs examined was not very large. I would expect a larger span to effect overall performance more significantly. The correlation between the $(\min + avg)/2$ aggregator and the balance of the model is undeniably strong. Every statistic examined, and the graphs plotted, clearly indicate that this fusion method strongly relates to the balance of the problem. At a minimum, this evidence should encourage modelers to consider fusion methods beyond the average when solving unbalanced classification problems. Hopefully it inspires some to look further into this relationship and further develop the underlying theory.

Table 6.8: Correlations between factors and yield.

	balance	AUC of good models	fraction good	number of models
(min+avg)/2 - avg	0.70	0.08	-0.17	-0.14
min - avg	0.58	0.16	0.06	-0.46

Paired t tests were also conducted for the differences between the (min + avg)/2 and the avg aggregator. This could also be viewed as a paired t between the x and y axis for each of the plots in figure 6.6. Each paired t test returned highly significant values ($\sim$ 1E-10 for each test). For 'balance' equal to 500 and 660, the average aggregator tested as superior. For the more highly imbalanced levels where 'balance' was 820 and 980, the (min + avg)/2 was undoubtedly superior.

6.7 Experimental Results with Several Datasets

This section includes empirical evidence for the previous discussion of synergistic classifier fusion. In addition to detailing results, this section also presents a method to create ensembles that will be referred to as the leave-l-features out ensemble method.

6.7.1 The leave-l-features-out Ensemble Method

Common problems in the security classification domain involve a large number of features and difficulty in feature selection. Feature selection is difficult because selecting salient features for a classification problem typically requires an adequate labeled sample of the positive and negative classes in training data. In the security classification domain, the positive class is a minority and often may not exist or may exist very sparsely in the training data. In either case, feature selection quickly becomes problematic. The solution proposed in this research involves using a subset of features in multiple models and then fusing these features to create a final decision value.

Consider the subspace creation technique for the security classification problem in this section as a *proposed* method for creating an ensemble. This is the same technique utilized in chapter 5. This section explicitly shows the technique in pseudo-code. Furthermore, considering again the practitioner, illustrating this technique shows how to bridge the gap between theory and application.

Feature selection is difficult in unbalanced problems, and this is a technique to reduce the effect of noisy features without identifying them. Explanation of underlying theory for the fusion technique used in this section will follow in later sections of this chapter, however the experimental results are presented first to create interest and perhaps some questions in the mind of the reader.

Again working in our security classification problem framework, it is assumed that a dataset, $\mathbf{X} \in \mathbb{R}^{N \times m}$, exists. $\mathbf{X}$ contains N instances or observations, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, where $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$. There are m variables to represent every instance. For every instance there is a label or class, $y_i \in \{-1, +1\}$. The unbalanced nature of this problem indicates a prevailing negative or healthy class and minimal instances of

the positive or unhealthy class.

The fusion method proposed first shuffles the ordering of the features in order to preserve randomness, and the next step builds a model from a subset of features. This method of building the ensemble is very similar in method to the leave one out model validation technique, except features instead of instances are left out and rather than leaving out one item, typically a small percentage (less than half) of the features is removed for each model in the ensemble. For example, if the leave out quantity (l) chosen is 10%, there will be ten models created, and each subset of features *left out* will be mutually exclusive however the features used to build each model will have significant overlap.

For example, suppose ten features, $\{a, b, c, d, e, f, g, h, i, j\}$ existed in a dataset. $m = 10$, and suppose we choose a leave out quantity of $l = 2$. The first step is to randomly order the features: $\{g, j, a, b, i, d, c, e, f, h\}$. Next, create $m/l = F = 5$ models with the following features:

$\{a, b, i, d, c, e, f, h\}$
$\{g, j, i, d, c, e, f, h\}$
$\{g, j, a, b, c, e, f, h\}$
$\{g, j, a, b, i, d, f, h\}$
$\{g, j, a, b, i, d, c, e\}$

These models would all be trained with the same data observations, each model built with the aforementioned leave out strategy. This leave out strategy creates $F = 5$ models each with different performance, and typically several of the models predict well and several are likely to perform poorly. Next, introduce the test data to each model and collect the results (decision values for each model). For each observation within each subspace, the classifier will produce a decision value, D_{ij}, where D_{ij} represents the decision value from the j^{th} classifier for the i^{th} observation. o_{ij} represents the ordinal position, or rank, of D_{ij} (for the same classifier, meaning j remains constant). For example, if D_{71} is the smallest value for the 1^{st} classifier, $o_{71} = 1$. The purpose of using the rank rather than the actual value, or scaled value, of the decision value hinges on the issue of fusing models. When fusing models it

is important that the variables which are being fused represent a similar unit of scale and magnitude. Essentially it is important to ensure that the fusion combines apples and apples rather than apples and oranges. The problem with decision values involves the underlying (and unknown) distributions of these values. It is possible to normalize the decision values from each model, however even then there is still influence from the parametrics of the underlying distribution. In order to get away from this "apples to oranges" problem, ranks are fused. Ranks eliminate parametric influence.

The algorithm describing this fusion method follows:

Algorithm 8 Fusing SCP Models

1: Select a number of features, l, to leave out from every model
2: Build $F = (m/l)$ (rounding up if m is not divisible by l) models
3: Map $D_{ij} \rightarrow o_{ij}$
4: **for** $i = 1$ to N **do**
5: fuse o_{ij} for $j = 1..F$
6: **end for**

6.7.2 Experimental Results

Initial experimental results with this fusion method explored a range of fusion methods, primarily focused on the max, avg, and min, and the spectrum spanning these functions. Four datasets were examined in this experiment. The Sick dataset was used twice with a different leave out value.

Dataset	m	N	p^*	l	**comment**
P300	100	4200	2100	10	Courtesy of Wadsworth Lab, www.wadsworth.org
Ionosphere	34	351	126	5	UCI Repository
Schonlau	26	5000	231	5	www.schonlau.net, [48–50]
Sick	137	5393	250	10	see ([76])
Sick	137	5393	250	50	see ([76])

Table 6.9: Datasets examined; m represents dimensionality, N represents no. of observations, l represents the leave out quantity. (* p represents the number of positive instances in the dataset.)

The learning model used for each dataset was the one-class SVM. Seven different fusion techniques were examined to include the *algebraic product*, min,

(min +*avg*)/2, *avg*, (max +*avg*)/2, max, *algebraic sum*. When fusing models with fuzzy logic aggregators (especially the *algebraic sum* and *algebraic product*), it is necessary to map the rank values into a range between 0 and 1. This is a simple scaling which can be done without the loss of any information. In order to measure the benefit of these fusion methods, the experiment involved 30 iterations of each fusion method, each with a different random training and test sample. The benchmark comparison for each fusion method was a one class SVM model built from all available variables in the dataset. This is referred to as the **base model** in the plots. Essentially the comparison involved whether the ensemble created with the fusion method improved performance over the benchmark model with all variables considered in one lump sum. The plots in figure 6.7 indicate performance of the fusion methods for each of these datasets.

Dataset	**algebraic product**	**min**	**(min+avg)/2**	**avg**	**(max+avg)/2**	**max**	**algebraic sum**
Ionosphere	NS	0.000000	0.000026	NS	NS	NS	NS
Schonlau	0.002224	0.038938	0.003518	0.009601	0.267514	NS	0.148508
Sick, $l = 50$	0.042059	NS	0.043812	0.629893	NS	NS	NS
Sick, $l = 10$	0.000065	0.279550	0.015050	0.004289	NS	NS	NS
P300	0.000000	0.004074	0.000000	0.000000	0.000000	0.000001	0.000000

Table 6.10: paired t-test values for the comparisons illustrated in Figure 6.7.

This experiment provided strong indications that certain fusion methods perform best. The paired t-test values in Table 6.7.2 indicate that the min fusion methods work best, with the function of (min+avg)/2 being the only function which illustrated a significant difference (improvement with fusion) for every dataset. A t-test value of 'NS' indicates that there was a difference in the mean of the experiments, however the direction of the difference was inverted. It indicated that the base model performed better. In order to eliminate confusion, an 'NS' indicates that the fused method was Not Significantly better than the base model.

Figure 6.7 provides insight into the behavior of various fusion methods. For each dataset, 30 different experiments were conducted. Each of these experiments consisted of a different random split of the data (typically a 50 / 50 split between training and test data for each dataset), and a different random shuffling of the

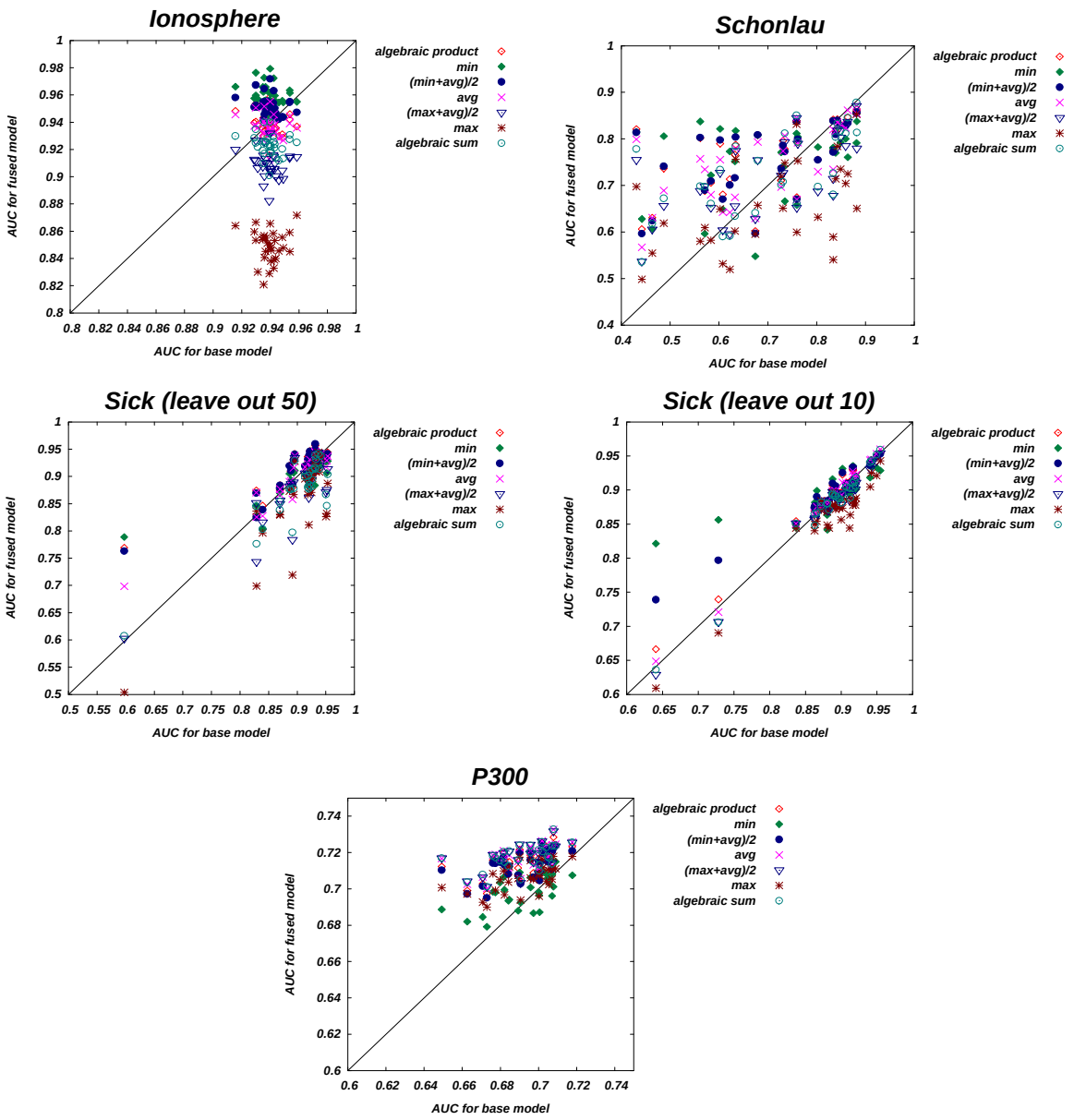

Figure 6.7: Synergistic fusion results for actual data

features in order to achieve different leave out ensembles (as per algorithm 8). The base model performance, shown on the horizontal axis, represents the performance of the one-class SVM with a linear kernel utilizing all variables. The vertical axis represents the performance of the fused models. Points plotted above the diagonal represent superior fused model performance. Points along the diagonal represent negligible differences in performance, and points below the diagonal illustrate supe-

rior performance of the base model. For every model except for the P300 data, the test data contained no more than 50% positive instances. The balance of positive instances contained within the data has a significant effect upon which fusion method works best. This will be explained in detail in the next section. Furthermore, it is apparent that the (min + avg) / 2 aggregator worked well for every model.

This experiment presents strong indications that certain fusion methods work best. The experiment alone cannot justify this claim, however coupled with the previous analysis of rank distributions and the effect of balance when fusing ranks, the argument for including the min when fusing ranks for unbalanced classification becomes convincing.

6.8 Conclusion

This chapter present several novel thrusts which present opportunities for improved ensembles as well as future directions for research with ensemble techniques.

Simulating rank distributions and creating pseudo ROC curves provide insight into the non-parametric statistics behind ROC curves and create number of analytical advantages. The insight provided by possessing control of the several parameters to include prediction power, balance of classes, and number of models enables analysis which is not possible when analyzing fusion metrics and ROC curves created from actual data. Analysis with actual data limits a researchers control of the parameters.

A critical advantage of this simulation analysis involves the convincing evidence created through simulation. Since the rank distributions and ROC curves are created from first principles and model generic, there is no concern of bias due to the characteristics of the data or behavior of a particular model. Results are general, and the general results of the simulation analysis provide a broader range of applicability for the research included in this chapter.

The final and perhaps most important finding in this chapter involves consideration of the min and max aggregators when fusing models. The avg aggregator has long been considered the best aggregator for fusion [18, 19, 74]. When comparing decision values or non-binary classification systems, the avg is essentially the

only metric considered. However, it is well known that there is a flaw of averages, including bias from outliers. The rank distributions studied in this chapter clearly illustrate that there are different likelihoods associated with balanced classification problems as opposed to unbalanced classification problems. The min or max aggregator capitalize on this difference in likelihoods and create improved results.

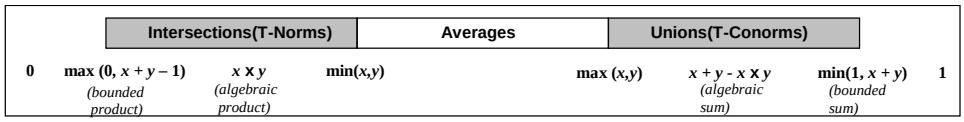

Figure 6.8: Spectrum of fuzzy aggregators.

Much opportunity for future research exists from the research proposed in this chapter. An analytical solution for the rank distributions would be a significant contribution. Additionally, this research only explores fusion metrics from one part of the aggregation spectrum. The fuzzy logic aggregation spectrum (see figure 6.8) encompasses the aggregators included in classic fuzzy logic, such as the the aggregators to the left of min and to the right of max, however it also includes the aggregation possibilities between the min and max. Fusion with $(\min + avg)/2$ is simply a heuristic that works well based upon theoretical understanding of the min and avg aggregators. However, it is possible that the midpoint between these aggregators is not appropriate. Perhaps it changes based upon the balance of the problem. Future research should consider these possibilities.

CHAPTER 7
SOME STATISTICAL PROPERTIES OF THE GAUSSIAN KERNEL MATRIX

Kernel-based pattern recognition has gained much popularity in machine learning and data mining because of proven performance and broad applicability [130]. Clustering, anomaly detection, classification, regression, and kernel based principal component analysis are just a few of the techniques that use kernels for some type of pattern recognition. The kernel is a critical component of these algorithms - arguably the most important component. There are many different kernels, however one of the most popular is the gaussian kernel. Machine learning researchers clearly favor the gaussian kernel as a top choice from all of the available kernels, and this is largely a result of the gaussian kernel's unique properties and detection of nonlinear patterns.

Before proceeding too far, let us clarify some notation. Let us assume that there is a given data set $\mathbf{X} \in \mathbb{R}^{N \times m}$. $\mathbf{X}$ contains N instances or observations, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, where $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$. There are m variables to represent each instance i. For every instance there is a label or class, $y_i \in \{-1, +1\}$. Equation 7.1 illustrates the formula to calculate a gaussian kernel.

$$\kappa(i, j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \tag{7.1}$$

A significant disadvantage for this kernel is the need to tune for the proper value of σ. This can be accomplished manually, through trial and error. An automated technique could involve stepping through a range of values for σ, perhaps in a gradient ascent optimization, seeking optimal performance of a model with training data. Regardless of the method utilized to find a proper value for σ, this type of model validation is common and necessary when using the gaussian kernel. Although this approach is feasible with supervised learning, it is much more difficult to tune σ for unsupervised learning methods. The one-class SVM, originally

proposed by Tax and Duin in [137] and also detailed by Scholkopf et. al. in [127], is a good example of an unsupervised learning algorithm where training validation is difficult due to the lack of positive instances. The one-class SVM trains with all negative instances or observations, and based upon the estimated support of the negative instances, new observations are classified as either inside the support (predicted negative instance) or outside of the support (predicted positive instance). It is quite possible, however, that there are very few or no positive instances available. This poses a validation problem.

The supervised learning approaches use validation based upon the availability of positive and negative classes of data. This validation requires the solving of multiple models, however it is also possible to explore direct kernel tuning for supervised learning as well. Direct tuning for supervised learning is briefly discussed in section 7.4.

There are several significant advantages of using an algorithm which automates the tuning of the gaussian kernel. The first advantage is the elimination of manual tuning, a tedious and often erroneous (due to the common mistake of overfitting) process. Another advantage is the potential for this algorithm to expand the use of gaussian kernels to many more models. Both [137] and [127] state that tuning the gaussian kernel for the one-class SVM is an open problem. The described automated tuning also introduces a fast method to include gaussian kernels within learning ensembles.

7.1 Recent Work

Tax and Duin [137] and Scholkopf et. al. [127] performed the groundbreaking work with the one-class SVM. Stolfo and Wang [132] successfully apply the one-class SVM to the intrusion data set that we use in this paper. Chen et. al. [27] uses the one-class SVM for image retrieval. Shawe-Taylor and Cristianini [130] provide the theoretical background for this method.

Tax and Duin [138] discuss selection of the σ parameter for the one-class SVM, selecting σ based upon a predefined error rate and desired fraction of support vectors. This requires solving the one-class SVM for various values of σ and the

parameter C, referred to in this paper as $1/\nu N = C$. This method relies on the fraction of support vectors as an indicator of future generalization. Tuning of the parameter C does not significantly impact the ordering of the decision values created by the one-class SVM; tuning of σ influences the shape of the decision function and profoundly impacts the ordering of the decision values. When seeking to maximize the area under the ROC curve (AUC), the ordering of the decision values is all that matters. The technique in this paper is very different from the one which is mentioned in [138]. We use the kernel matrix directly, therefore the one-class SVM does not need to be solved for each change in value of σ. Furthermore, tuning the kernel matrix directly requires the tuning of only one parameter.

The other outcome of this research involves further insight into the statistics of this kernel. The gaussian kernel has special properties, and the squared coefficient of variance, introduced in this chapter, illustrates some of these special properties. Improved understanding of the statistical behavior of this kernel opens possibilities to improved applications and better insight into how and why this kernel performs well.

7.2 The One-Class SVM

The one-class SVM is an anomaly detection model solved by the following optimization problem:

$$\min_{R\in\mathbb{R},\zeta\in\mathbb{R}^N,c\in F} \quad R^2 + \frac{1}{\upsilon N}\sum_i \zeta_i \tag{7.2}$$

$$\text{subject to} \quad \| \Phi(\mathbf{x}_i) - c \|^2 \leq R^2 + \zeta_i \text{ and } \zeta_i \geq 0 \text{ for } i = 1, ..., N$$

The lagrangian dual is shown below in equation 7.3.

$$\max_{\alpha} \sum_i \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_i) - \sum_{i,j} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{7.3}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{vN} \text{ and } \sum_i \alpha_i = 1$$

Scholkopf et. al. point out the following reduction of the dual formulation when modeling with gaussian kernels:

$$\min_{\alpha} \sum_{i,j} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{7.4}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{vN} \text{ and } \sum_i \alpha_i = 1$$

This reduction occurs since we know that $\kappa(\mathbf{x}_i, \mathbf{x}_i) = 1$ and $\sum_i \alpha_i = 1$. Equation 7.4 can also be written as $\min \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$. Shawe-Taylor and Cristianini [130] explain that $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$ is the weight vector norm, and controlling the size of this value improves the statistical stability, or regularization of the model.

All training examples with $\alpha_i > 0$ are support vectors, and the examples which also have a strict inequality of $\alpha_i < \frac{1}{vN}$ are considered non-bounded support vectors.

In order to classify a new test instance, $\mathbf{v}$, we would evaluate the following decision function:

$$f(v) = \kappa(\mathbf{v}, \mathbf{v}) - 2\sum_j \alpha_j \kappa(\mathbf{v}, \mathbf{x}_j) + \sum_{j,k} \alpha_k \alpha_j \kappa(\mathbf{x}_k, \mathbf{x}_j) - R^2$$

Before evaluating for a new point, R^2 must be found. This is done by finding a non-bounded support vector training example and setting the decision function equal to 0 as detailed by Bennett and Campbell in [8]. If the decision function is negative for a new test instance, this indicates a negative or healthy prediction. A positive evaluation is an unhealthy or positive prediction, and the magnitude of the decision function in either direction is an indication of the model's confidence.

7.3 Method

The behavior of the gaussian kernel is apparent when examined in detail. The values lie within the (0,1) interval. A gaussian kernel matrix will have ones along the diagonal (because $\| \mathbf{x}_i - \mathbf{x}_i \| = 0$). Additionally, a value too small for σ will force the matrix entries towards 0, and a value too large for σ will force matrix entries

towards 1.

There is also a property of all kernels, which we will refer to as the fundamental premise of pattern recognition, which simply indicates that for good models, the following relationship consistently holds true:

$$(\kappa(i,j)|(y_i = y_j)) > (\kappa(i,j)|(y_i \neq y_j)) \tag{7.5}$$

Consistent performance and generalization of the fundamental premise of pattern recognition is the goal of all kernel based learning. Given a supervised dataset, a training and validation split of the data is often used to tune a model which seems to consistently observe the fundamental premise. However, in an unsupervised learning scenario positive labeled data is limited or non-existent, and furthermore, models such as the one-class SVM have no use for positive labeled data in the training data.

A first approach towards tuning a kernel matrix for the one-class SVM might lead one to believe that the matrix should take on very high values, indicating that all of the kernel entries for the training data is of one class and therefore should take on high values. Although this approach would first seem to be consistent with the fundamental premise in equation 7.5, this approach would be misguided. The magnitude of the values within the kernel matrix is not an important attribute. The important attribute is actually the spread or the variance of the entries in the kernel matrix. At first this may seem to be anomalous with equation 7.5, however a closer examination of the statistics of a kernel matrix illustrates why the variance of the kernel matrix is such a critical element in model performance.

Shawe-Taylor and Cristianini point out that small values of σ allow classifiers to fit any set of labels, and therefore overfitting occurs [130]. This phenomenon is shown later in figures 7.4 and 7.5. They also state that large values for σ impede a classifiers ability to detect non-trivial patterns because the kernel gradually reduces to a constant function. The following mathematical discussion supports these comments for the one-class SVM. Considering again the one-class SVM optimization problem, posed as $\min \boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$. Assuming use of the gaussian kernel, if the sigma parameter is too small and $\kappa(i,j) \rightarrow 0$, the optimal solution is $\alpha_i = 1/N$. Equation 7.4, the objective function, will equate to $1/N$ (since $\sum_i (1/N)^2 = 1/N$). If

the sigma parameter is too large and $\kappa(i,j) \rightarrow 1$, the optimal solution is the entire feasible set for $\boldsymbol{\alpha}$. Given these values for the variables and parameters, the objective function will now equate to 1. The brief derivation for the case when $\kappa(i,j) \rightarrow 1$ follows:

$$\alpha' \mathbf{K} \alpha = \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} \sum_{j=i+1}^{N} 2\alpha_i \alpha_j \kappa(i,j)$$

$$= \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} \alpha_i \sum_{j=1 \ldots i-1, i+1 \ldots N} \alpha_j$$

$$= \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} \alpha_i (1 - \alpha_i) = \sum_{i=1}^{N} \alpha_i^2 + \sum_{i=1}^{N} \alpha_i - \sum_{i=1}^{N} \alpha_i^2 = \sum_{i=1}^{N} \alpha_i = 1$$

The objective function bounds are $(1/N, 1)$, and the choice of σ greatly influences where in this bound the solution lies.

7.3.1 The Squared Coefficient of Variance

In order to find the best value for σ, a heuristic is employed. This heuristic takes advantage of the behavior of the one-class SVM when using the gaussian kernel. The mean and the variance of the non-diagonal kernel entries, $\kappa(i,j)|i \neq j$, play a crucial role in this heuristic. We will refer to the mean as $\bar{\kappa}$ and the variance as s^2. For any kernel matrix where $i, j \in \{1, ..., N\}$, there are $N^2 - N$ off diagonal kernel entries. Furthermore, since all kernel matrices are symmetrical, either the upper or lower diagonal entries only need to be stored in memory, of which there are $l = (N^2 - N)/2$. From here forward, the number of unique off diagonal kernel entries will be referred to as l.

It is first necessary to understand the statistic used in this heuristic, the coefficient of variance. The coefficient of variance is commonly referred to as 100 times the sample standard deviation divided by its mean, or $\frac{100s}{\bar{x}}$. This statistic describes the relative spread of a distribution, regardless of the unit of scale. Due to the scale and behavior of $\bar{\kappa}$ and s, this coefficient of variance monotonically increases for gaussian kernels as σ ranges from 0 to ∞. Using the sample variance rather than the standard deviation, different behavior occurs. The monotonic increase of

the coefficient of variance occurs because when σ is small, $s > \bar{\kappa}$; however, as σ increases, there is a cross-over point and then $s < \bar{\kappa}$. However, the variance of $\kappa(i,j)|i \neq j$ is always smaller than the mean of $\kappa(i,j)|i \neq j$. This property is what makes the variance of a kernel matrix such a critical component for the direct tuning method. The proof follows. For the sake of notation simplicity, $x_k \in (0,1)$, $k \in [l]$, will represent off diagonal kernel entries, that is entries $\kappa(i,j)|i \neq j$.

$$VAR(x_k) \leq \bar{x} \implies \frac{\sum_k x_k^2 - 2\bar{x}\sum_k x_k + l\bar{x}^2}{l-1} \leq \frac{(l-1)\bar{x}}{l-1}$$

$$\implies \frac{l\bar{x}^2 - 2l\bar{x}^2 - (l-1)\bar{x} + \sum_k {x_k}^2}{l-1} \leq 0$$

$$\implies \sum_k x_k^2 - l\bar{x}^2 - (l-1)\bar{x} \leq 0 \implies \sum_k x_k^2 - \sum_k x_k + \frac{\sum_k x_k}{l}(1 - \sum_k x_k) \leq 0$$

$$\implies l\sum_k x_k^2 - \sum_k x_k(l-1) - (\sum_k x_k)^2 \leq 0$$

$$\implies \sum_k x_k^2 - (\sum_k x_k)^2 + (l-1)\sum_k x_k^2 - (l-1)\sum_k x_k \leq 0$$

$$\implies \underbrace{\sum_k x_k^2 - (\sum_k x_k)^2}_{\text{always } \leq 0} + (l-1)\underbrace{\left(\sum_k x_k^2 - \sum_k x_k\right)}_{\text{always } \leq 0 \text{ for } 0 \leq x_k \leq 1} \leq 0$$

The fact that the variance of $\kappa(i,j)|i \neq j$ is always smaller than the mean of $\kappa(i,j)|i \neq j$ indicates that the squared coefficient of variance, $s^2/\bar{\kappa}$, is a fraction. Furthermore, as σ ranges from 0 to ∞, this fraction ascends to a global max. In order to protect against division by zero and round off error, a small value, ϵ, can be added to the denominator.

The results in the following objective function for optimization which can be solved quickly with a gradient ascent algorithm. Solving the optimization problem in equation 7.6 leads to the best choice for σ.

$$\max_{\sigma} \frac{s^2}{\bar{\kappa} + \epsilon} \; \forall (i \neq j) \tag{7.6}$$

such that

$$\kappa(i,j) = e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}} \; , \; \bar{\kappa} = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} \kappa(i,j)}{l} \; , \; s^2 = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} (\kappa(i,j) - \bar{\kappa})^2}{l-1}$$

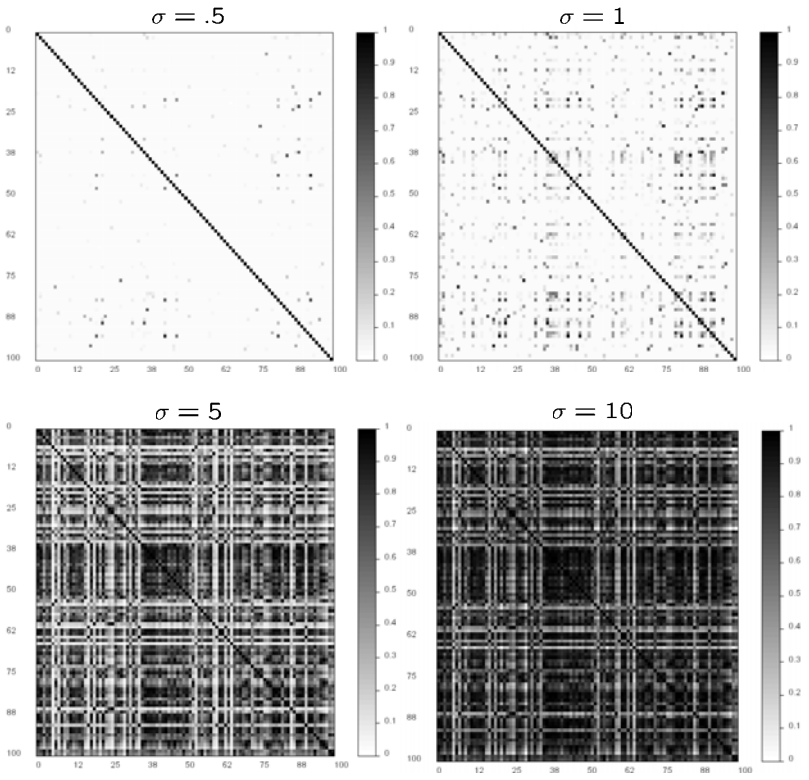

Figure 7.1: Color visualization of a kernel matrix with various values for σ. This visualization involves 100 observations where kernel entries close to 0 are light; darker entries represent values closer to unity.

Figure 7.1 illustrates the impact of sigma on the kernel matrix. The dataset used to create figure 7.1 was the ionosphere data which is available from the UCI repository. These data are all of the negative class, and it is evident that both the kernel element values and the dispersion of these values changes as σ changes. The

optimal value for σ for this dataset is 1, and the color visualization of the kernel matrix when $\sigma = 1$ clearly indicates that there is dispersion in the kernel matrix. However, it is also noticeable that the central tendency of the kernel entries for this optimal σ is a small value, closer to zero than one. This is consistent for all of the datasets. This is why it is important to use a metric that detects relative dispersion and is not biased by the magnitude of the kernel entries.

7.3.2 Visualization of the One-Class SVM

It is useful to visualize the decision function of the one class SVM with a small two dimensional dataset. The dataset utilized for this experiment is a toy dataset created by the author and shown in figure 7.2. The points were generated from a uniform distribution and manipulated to stay close to the horizontal and vertical axis.

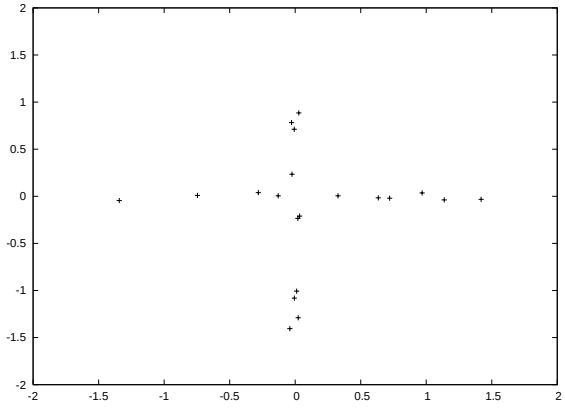

Figure 7.2: Toy two dimensional dataset - the cross data

Figure 7.3 illustrates the behavior of the gaussian kernel for the two dimensional cross data. Notice that the squared coefficient of variance, $s^2/(\bar{\kappa} + \epsilon)$ peaks for $\sigma = .1$. This will prove meaningful after reading the next section of this chapter.

Figures 7.4 and 7.5 illustrate how the decision function of the one class SVM changes as σ increases across the same spectrum as many of the plots in this chapter. These plots were created with the PM3D function in gnuplot 4.0 [140].

The plots corresponding to a small σ value clearly illustrate that overtraining is occurring, with the decision function wrapped tightly around the data points. On

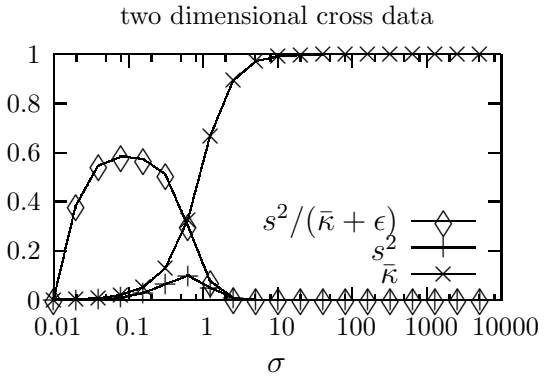

Figure 7.3: Gaussian kernel statistics for the two dimensional cross data

the other hand, large values of sigma simply draw a tight oval around the points without defining the shape or pattern. The best value for sigma lies somewhere in between. This chapter is an effort to better define where that somewhere in between lies.

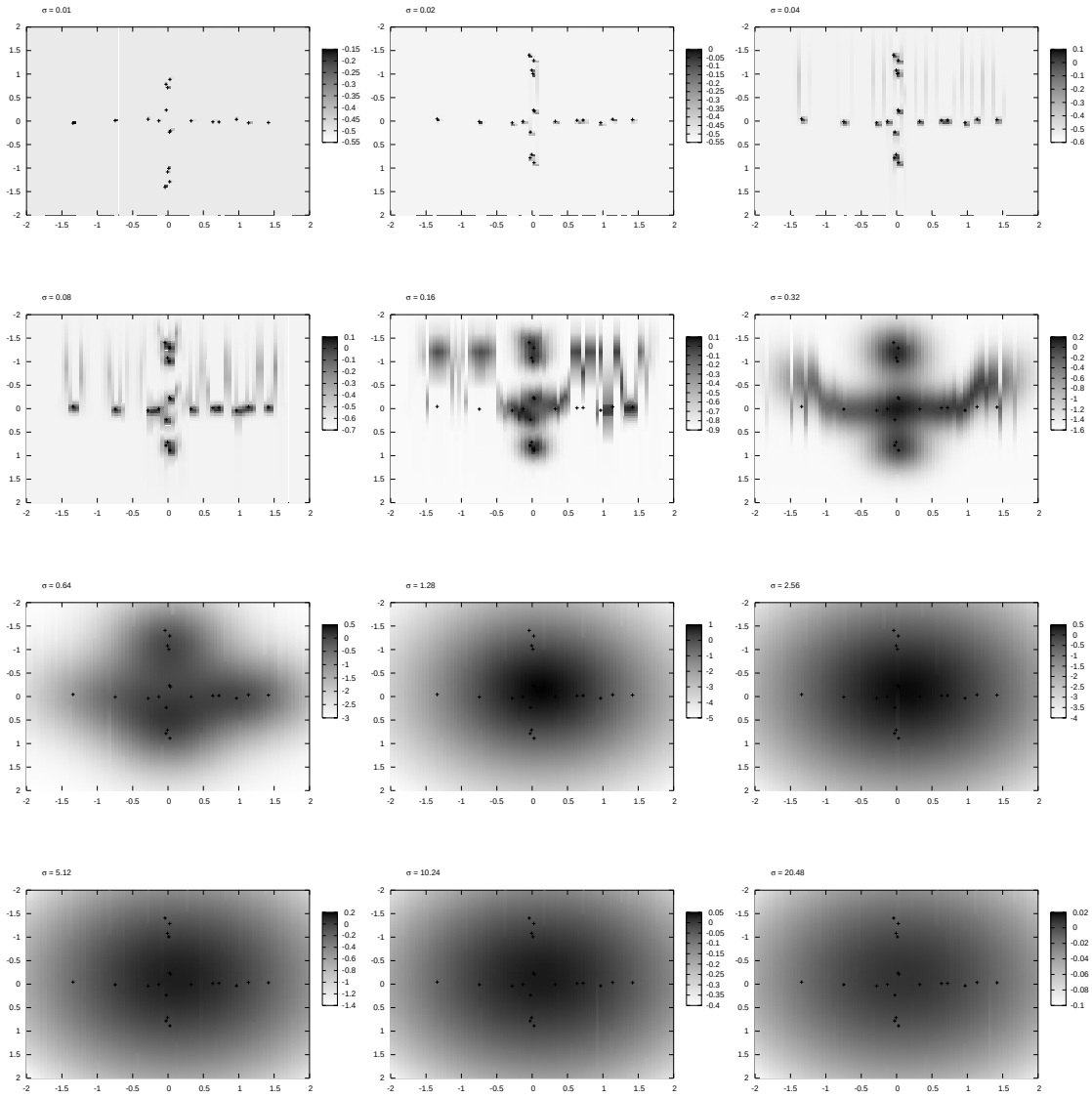

Figure 7.4: Visualization of one class SVM for smaller values of σ

7.3.3 Why Direct Tuning of the Kernel Matrix Works

As mentioned previously, it is desirable to minimize $\boldsymbol{\alpha}'\mathbf{K}\boldsymbol{\alpha}$. $\boldsymbol{\alpha}$ is sparse when there are few support vectors, and this sparseness is typically desirable to minimize complexity and improve regularization. However, meaningless sparse solutions for α can occur as all $\kappa(i,j) \rightarrow 0$. Meaningful sparse solutions for $\boldsymbol{\alpha}$ occur when the kernel matrix entries are not concentrated either towards 0 or 1, but are showing good dispersion and there is payoff in the optimization to select the few instances

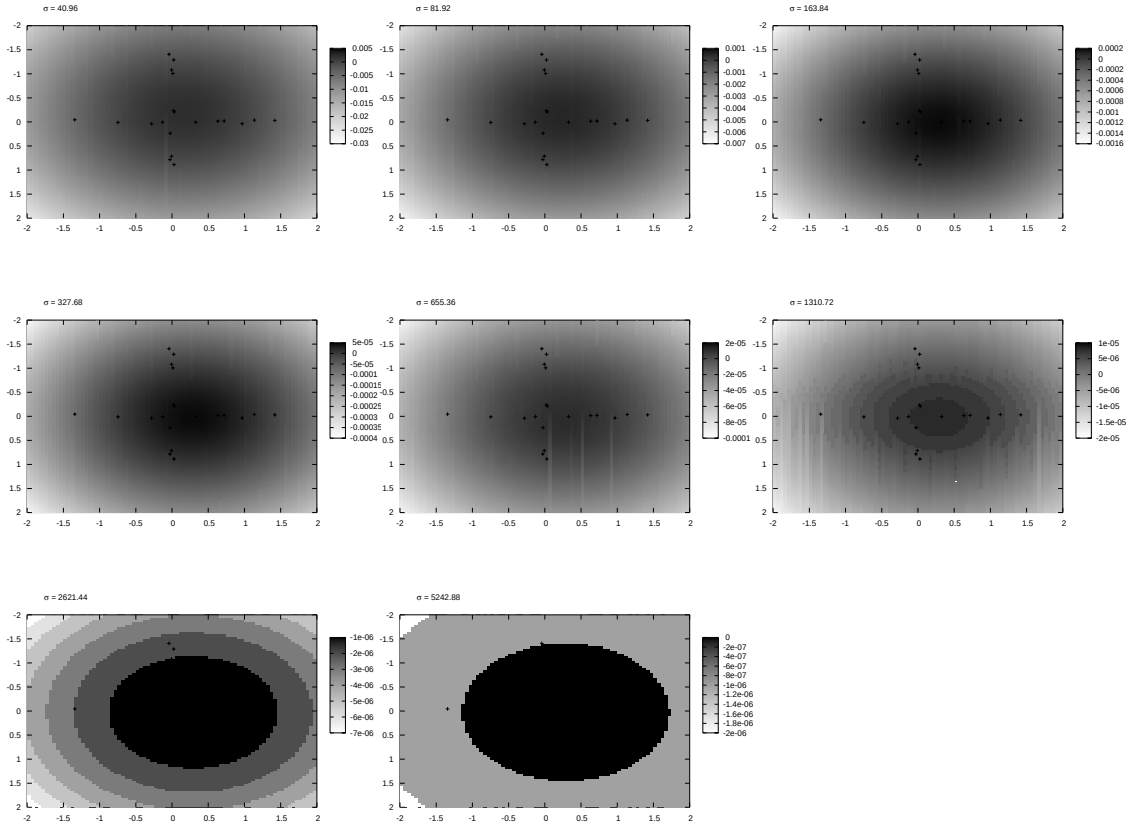

Figure 7.5: Visualization of one class SVM for larger values of σ

which clearly define the margin of the density approximated by the one-class SVM. Although the variance, s^2, is a good indicator of the dispersion in the kernel matrix, it is biased towards a larger σ since variance is affected by unit of scale or magnitude. $s^2/(\bar{\kappa}+\epsilon)$ is robust against unit of scale. This statistic illustrates the dispersion of a distribution regardless of scale. For the one-class SVM, the optimal value for σ and maximum value for $s^2/(\bar{\kappa}+\epsilon)$ typically occurs as σ increases from 0 and the kernel entries first begin to illustrate dispersion. When there is maximal dispersion in the kernel matrix, the values assigned to $\boldsymbol{\alpha}$ will reflect the behavior and relationships of observations, which is the purpose of the statistical learning. Maximal dispersion of the kernel matrix also supports the fundamental premise of pattern recognition. When σ is not properly tuned, the values assigned to $\boldsymbol{\alpha}$ will be erroneously based upon the behavior of the optimization problem or the optimization algorithm used to solve the problem.

7.4 Comments on Supervised Learning

A similar approach can be taken for the supervised binary classification problem. Recall that the fundamental premise of pattern recognition states that for all good models, $(\kappa(i,j)|(y_i = y_j)) > (\kappa(i,j)|(y_i \neq y_j))$. Therefore, if we can achieve maximum clarity on the distribution of $(\kappa(i,j)|(y_i = y_j)) - (\kappa(i,j)|(y_i \neq y_j))$, optimal model performance should occur. This maximal clarity occurs when there is dispersion in this distribution, again by examining a metric inspired from the coefficient of variance and the unsupervised direct tuning method.

$$\max \frac{s^2_{y_i=y_j} + s^2_{y_i \neq y_j}}{\bar{\kappa}_{y_i=y_j} - \bar{\kappa}_{y_i \neq y_j} + \epsilon}$$

This optimization only considers the case when $\bar{\kappa}_{y_i=y_j} - \bar{\kappa}_{y_i \neq y_j} > 0$, meaning that the two extremes of the kernel tuning spectrum are simply eliminated. This method focuses on the difference between similar kernel elements and dissimilar kernel elements. This is essentially the margin, or the boundary that defines the separation of the two classes. Achieving clarity on this margin is the key to tuning for supervised learning. Unlike unsupervised learning, validation is possible with supervised learning. It is entirely possible to automate tuning in a supervised setting using validation to tune the kernel. If validation is possible, it should be performed. The heuristic shown above could be an additional measure used to check the behavior of the kernel for supervised learning, however it is always prudent to use model validation if it is possible.

7.5 Experimental Results

In order to evaluate the performance of the direct tuning heuristic, several experiments were conducted. The data included three benchmark sets: banana, chessboard, and ionosphere (from UCI repository). Additionally, two computer intrusion datasets named Schonlau and Sick, after their respective creators, are also examined. The Schonlau data involves determining authenticity of a user based on UNIX commands. This data was originally discussed in [41, 42, 129] and the actual data used in this paper was also discussed in [48–50]. The Sick data was originally

examined in [76]. The parameter ν is set to .5 for every experiment.

Dataset	dimensions	positive	negative	comment
Banana	2	50	50	see [62]
Chessboard	2	50	50	www.cs.wisc.edu/ ∼olvi/data/check1.txt
Ionosphere	34	126	225	UCI Repository
Schonlau	26	231	4769	www.schonlau.net
Sick	137	250	5143	see [76]

For each dataset, one half of the negative class was used for training the one-class SVM and the test data comprised of the other half of the negative class and all members of the positive class. The same split remained consistent for all experiments for the purposes of control and consistency. The performance measure utilized is the area under the receiver operating characteristic curve (AUC).

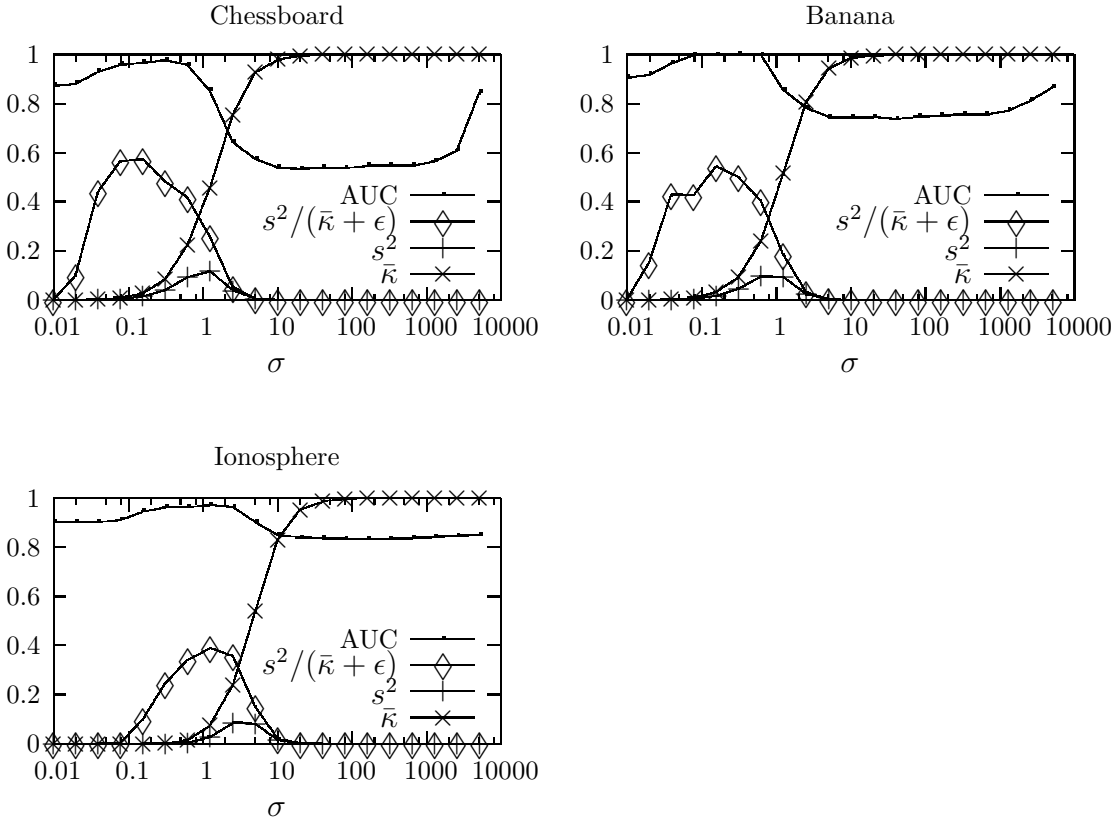

Figure 7.6: Experimental results for three benchmark datasets

The three benchmark datasets clearly illustrated optimal performance when $s^2/(\bar{\kappa}+\epsilon)$ is optimal. For each of the benchmark solutions, a gradient ascent algorithm was tested and converged on the optimal σ within 4-6 iterations depending on the initial values and dataset. An initial value of 1 for σ seemed to work well since most of the optimal values for σ ranged between .1 and 10.

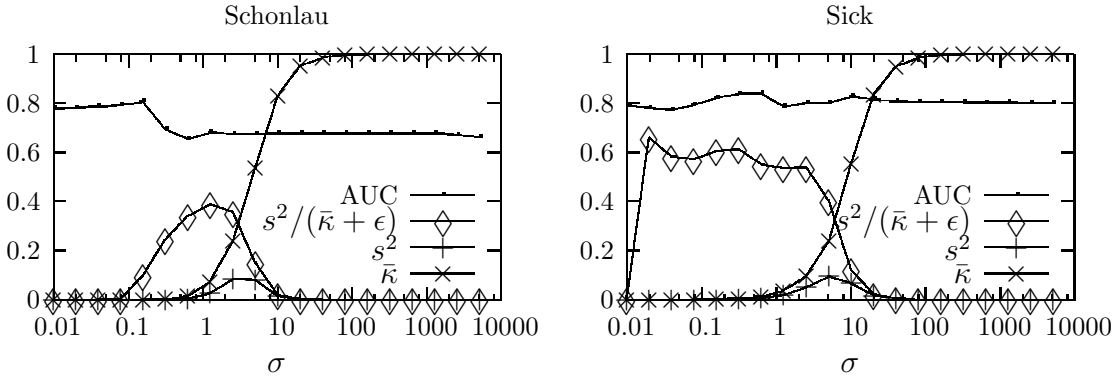

Figure 7.7: Experimental results of two computer intrusion datasets

The two computer intrusion datasets did not clearly indicate this same type of ideal performance. The Schonlau dataset performed the worst by far, and the Sick dataset did indicate a region of good performance. The value of σ seemed indifferent for the Sick data. σ tuning will always be a data dependent heuristic, and it is quite possible that sigma tuning with these datasets requires robust validation in order to discover the sensitive σ window where optimal performance occurs. However, anytime that the dataset requires such robust validation to uncover a sensitive σ window, researchers should exercise caution. This is an indication that overtraining is taking place, meaning that the function to classify this data has potentially found a spurious pattern within the data that may not generalize.

This can be explained by taking a closer look at the behavior of the squared coefficient of variance as dimensionality increases. Dimensionality is a fundamental problem with the security classification problem. Chapter 1 briefly introduces the problem of dimensionality with kernel based learning, specifically illustrating the impact of dimensionality on the gaussian kernel assuming normality of the underlying variables. The gaussian kernel, $\kappa(i,j) = e^{\frac{-\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}}$, compares two vectors

or observations utilizing the euclidean distance squared, $\| \mathbf{x}_i - \mathbf{x}_j \|^2$. It is well known that as dimensionality grows, points within the space approach becoming equidistant [93]. This indicates that it becomes more difficult distinguish similar from dissimilar points utilizing a distance metric such as the euclidean distance. The gaussian kernel transforms this distance in a non-linear fashion, however this distinguishability problem manifests itself through the sensitivity of the kernel to variations in σ. The squared coefficient of variance exposes this sensitivity.

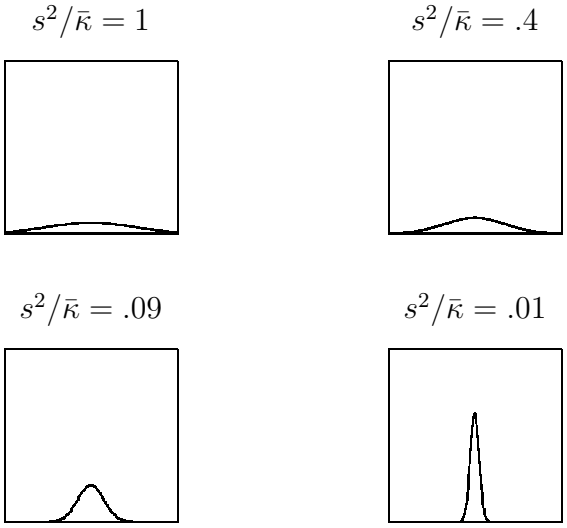

Figure 7.8: Four normal distributions with different squared coefficients of variance

Consider the following example. Figure 7.8 illustrates four different normal distributions with various squared coefficients of variance. An initial observation of these plots could lead one to question whether or not there is a difference between the coefficient of variance and the variance of the distribution. However, it is entirely possible that the variances of each one of those distributions are equal. Variance is relative, however the coefficient of variance is not. The gaussian kernel distribution is an excellent distribution to analyze with the squared coefficient of variance. It is a non-zero distribution, preventing false conclusions that could occur with a distribution centered on or near zero. Notice that the axis are not labeled in Figure 7.8. This is intentional. There is a difference that seems subtle at first, however when considered in the context of the gaussian kernel it becomes more obvious.

The squared coefficient of variance is a ratio and therefore not influenced by the magnitude of the random variable. Variance is influenced by the magnitude of the random variable. The squared coefficient of variance (or the coefficient of variance) is an overall indicator of the dispersion of a distribution, regardless of the magnitude of the values the random variable assumes. This indifference to magnitude is critical with the gaussian kernel since $\kappa(i,j) \in (0,1)$, with values typically clustering at the extremes for most values of σ. The experiments in this chapter indicate that the best values for σ often occur when the kernel values are small. The actual variance of the kernel for these values of σ would be a small number, however the squared coefficient of variance is largest in this region illustrating a large dispersion of kernel values.

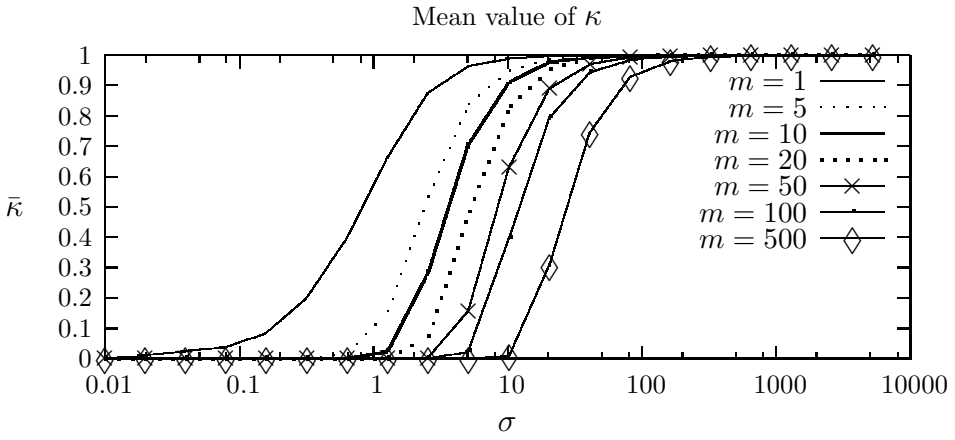

Figure 7.9: Mean value for gaussian kernels with various values of σ

Figures 7.9 and 7.10 illustrate the behavior of the kernel as σ changes. Figure 7.10 illustrates the behavior of the squared coefficient of variance of the gaussian kernel as σ changes for dimensionalities (m) ranging from 1 to 500. The squared coefficient of variance is a smaller value as dimensionality grows. The kernel is more likely to cluster at the extremes, and there is a small range of σ where transition occurs between the extremes.

The distance between points in a large dimensional space will be a random variable with a small squared coefficient of variance, indicating that in a large dimensional space points become equidistant. This problem with dimensionality could

explain why the Schonlau and Sick datasets did not perform well. The ionosphere data also contains quite a few dimensions, however many of these dimensions contain very little variance. Some of the features are highly imbalanced and binary, contributing very little to a dimensionality problem.

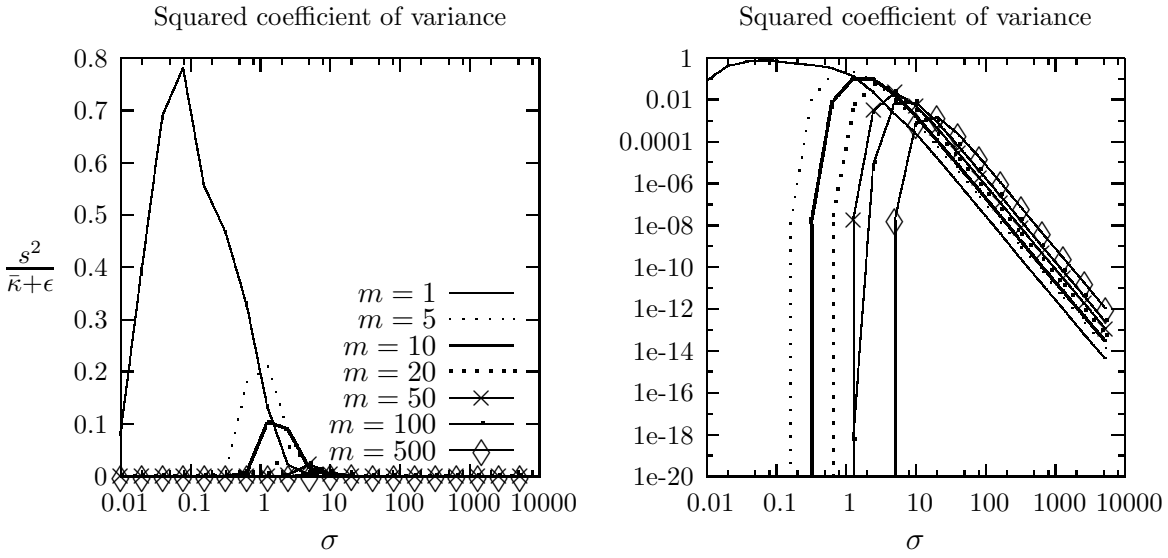

Figure 7.10: Squarred coefficient of variance for gaussian kernels with various values of σ

Figure 7.10 illustrates the behavior of the squared coefficient of variance of the gaussian kernel as σ changes for dimensionalities (m) ranging from 1 to 500. The squared coefficient of variance is a smaller value as dimensionality grows. The kernel is more likely to cluster at the extremes, and there is a small range of σ where transition occurs between the extremes. This behavior of the gaussian kernel for high dimensional spaces explains why the gaussian kernel tends to be sensitive with only a small viable range for σ.

The final experiment of this chapter involved exploring whether or not reducing dimensionality for the Schonlau and Sick datasets would improve the performance of the squared coefficient of variance heuristic. Figures 7.11 and 7.12 illustrate the performance of the squared coefficient of variance heuristic as dimensionality reduced through principal components. The AUC of both datasets degrade very little, if at all, as fewer principal components are utilized to represent the data. It

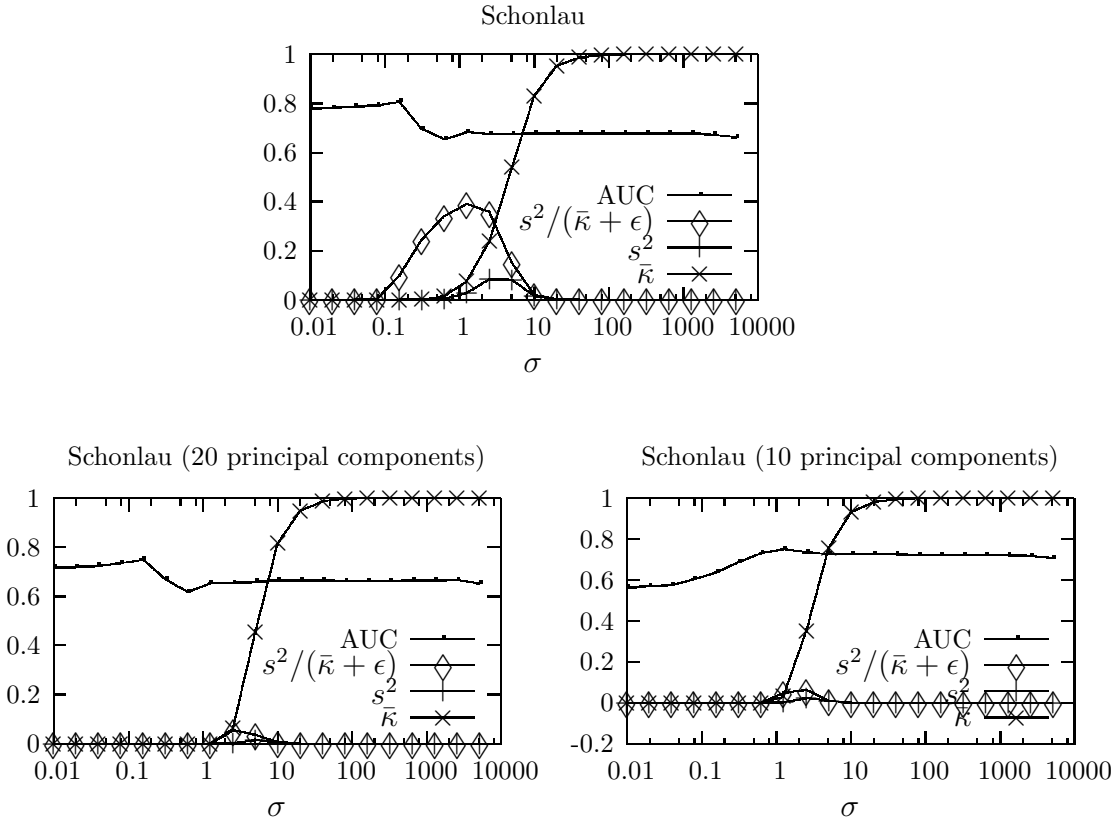

Figure 7.11: Schonlau results after principal component reduction

is well known that principal components reduce noise and capture the essence of a multidimensional space, and it is quite possible that this dimensionality reduction serves to stabilize the classifier across the spectrum of σ values and reduce the likelihood of overtraining or finding spurious patterns. This statement, however, is simply an unsupported hypothesis that requires further investigation. The purpose of the experiments summarized in figures 7.11 and 7.12 involved supporting the previous discussion which indicated that the squared coefficient of variance coefficient performs well in a smaller dimensional space.

7.6 Conclusions

Direct tuning of the gaussian kernel matrix is a novel and promising approach. The squared coefficient of variance heuristic proposed is grounded and supported with underlying theory. The study of this statistic enhances understanding of the

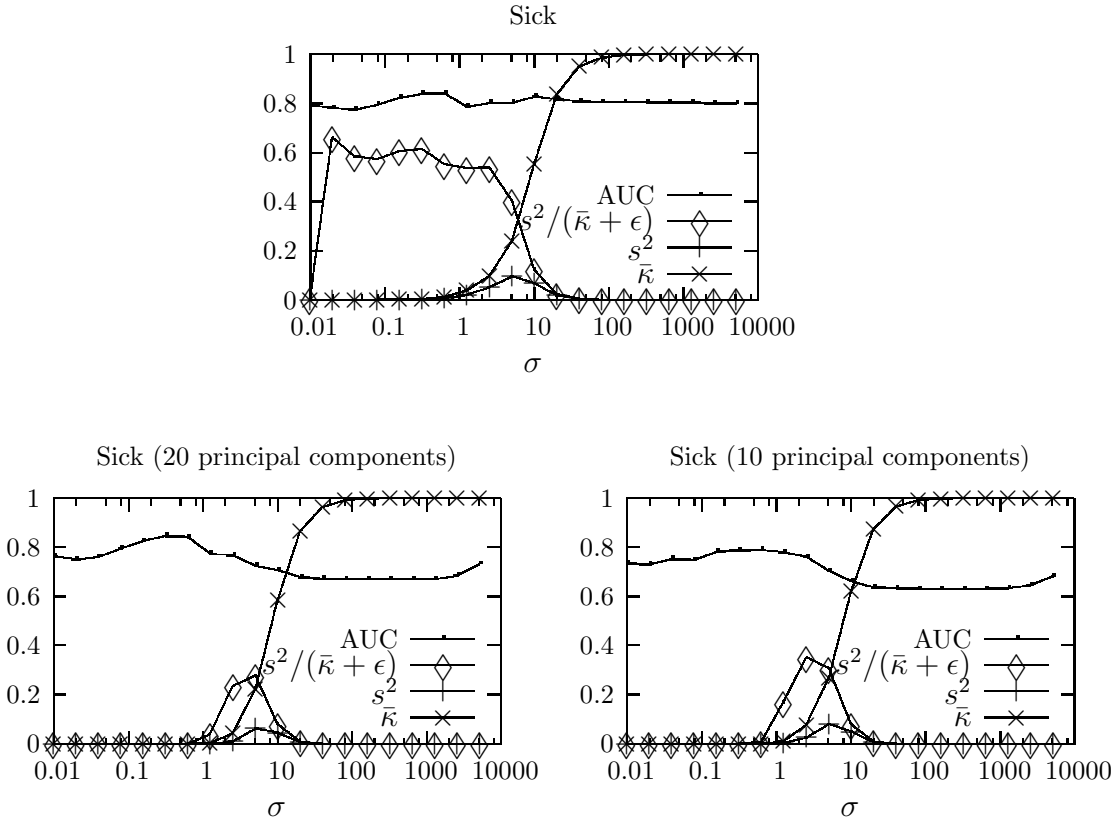

Figure 7.12: Sick results after principal component reduction

gaussian kernel, exposing special properties of this kernel and linking these special properties to classifier performance.

The significance of tuning the gaussian kernel for unsupervised learning applies directly to ensemble methods. Techniques such as bagging [18], random subspace selection [74], and fuzzy aggregation techniques [48, 49] can be employed for unsupervised learning with the gaussian kernel.

Future research could involve automated tuning approaches for supervised learning, however this approach will clearly have to outperform the traditional technique of validation. Direct tuning for supervised learning would be faster, but first it needs to be shown that it is also just as accurate as validation.

CHAPTER 8
THREE APPLIED CASES

8.1 Introduction

During the course of the research discussed in this dissertation, several datasets have been used to illustrate theory and heuristics. Additionally, there were three different case studies explored that demonstrate security classification problem applications. The typical classification problem framework requires instances represented as vectors, with each instance classified in a positive or negative class. The security classification problem (SCP) introduced in this dissertation requires further that the instances are unbalanced, a typical characteristic of security environments. All three case studies discussed in this chapter represent SCPs. During the course of discussing each case, practical comments will be included which highlight the process behind converting the data from its raw format to a usable machine learning format. This preprocessing is often the most difficult aspect of this type of work. Efficient and meaningful preprocessing is largely a product of experience, and this chapter aims to share some of that experience. The three case studies explored in this chapter, all SCPs, consist of two military applications and one network security application.

- Skaion Dataset. The Skaion Corporation donated a sample of simulated network data for the sake of the research in this dissertation. This network data consisted of TCPdump data (approximately 28,000 connections). The data contains numerous automated attacks, all of which have been identified with *Snort* [2]. This case illustrates that machine learning applications easily replicate the performance of signature based detection, leading us to the argument that machine learning applications could replace or minimally augment signature based detection based upon their ability to detect new attacks which are slight variations of previous attacks. Signature based detections cannot detect these types of subtle changes whereas machine learning applications provide potential to detect similar signatures or patterns.

- Geospatial Datamining for Threat Templating. This application involves prediction modeling of military threat incidents based upon geospatial statistics. The geospatial statistics measure the distance from the incident location to specific terrain features with the underlying assumption that incidents occur with a similar terrain feature proximity pattern. It is possible that this pattern exists beneath even the awareness of the threat.

- Insights into Insurgent Response to Traffic Flow Strategies. This effort involved simulating the effect of different traffic control point (TCP) strategies on a vehicle born improvised explosive device (VBIED). The SCP application involved with this application included a data farming exercise which explored a broad parameter space in an attempt to discover patterns which consistently succeeded or failed. This was a particularly ground breaking effort which involved a novel application (effect of TCP strategies versus VBIED success), unique computing techniques, and finally assessment which applied techniques which are a novel product of this dissertation. A detailed account of this work has been published in [66].

8.2 Skaion Dataset

The Skaion dataset provided an opportunity to analyze TCPdump data containing labeled attacks previously identified with *Snort* signatures. The purpose of this case study involved validating previously discussed network data connection quantification (utilizing TCPtrace and Perl programs) and exploring the feasibility of using machine learning applications to learn and detect attacks identified by signature based detection schemes.

This study exposed several useful insights:

1. Quantification of network traffic and labeling of network traffic connections for the purpose of machine learning is possible and relevant for network security.

2. Color visualization of large, high dimensional datasets is a useful and newly viable method to illustrate linear patterns within the data.

3. Machine learning applications closely replicate signature based detection and present potential to detect new security vulnerabilities which may be detected

due to their similarity to recognized signatures.

In a simple yet exploratory effort, quantification of network traffic connections utilizing TCPtrace [116] provided a platform to build a bridge between network data and machine learning. TCPtrace quantifies and aggregates network traffic based upon connections. TCPtrace provides a summary which can be parsed for further processing. The Skaion data contained unique IP addresses for the attacks identified by *Snort*, and therefore labeling connections based upon IP addresses was a straight forward match and parse operation with Perl. Figure 8.1 illustrates this process as well as providing an example of TCPdump and TCPtrace output.

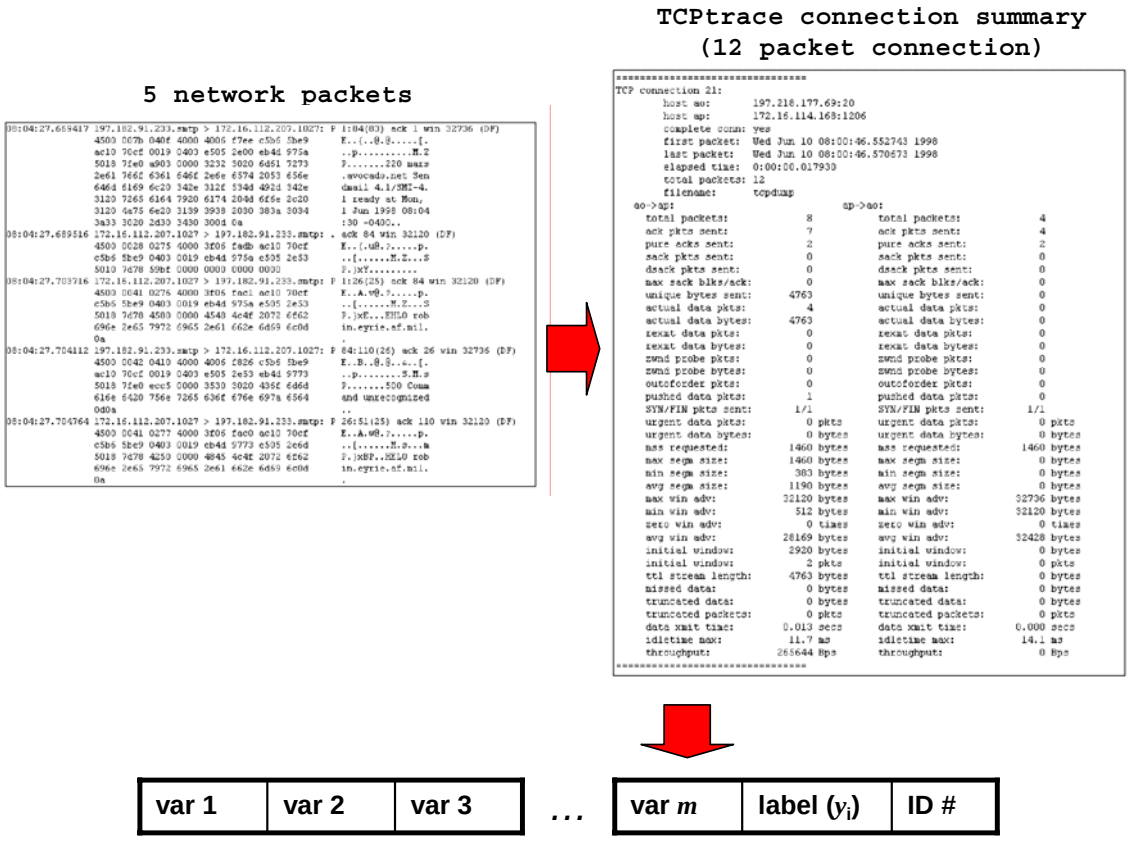

Figure 8.1: Conversion of TCPdump traffic to vector summaries of connections

Once the preprocessing depicted in figure 8.1 this network security problem becomes a machine learning problem. TCPtrace creates over 60 variables for statistical learning. This particular application contained two types of attacks: background scanners and background attackers. Table 8.1 is a description of the attacks

contained in the Skaion dataset.

Table 8.1: Skaion Data

Exploit Name	**Short description**	**Rate observed (attacks/day)**
loopback_web	Malformed loopback packet web server scan	275.5
openssl_too_open	OpenSSL KEY-ARG overflow exploit	24
opensslapache_bug	Apache OpenSSL bug exploit	112
ssh_crc32	OpenSSH CRC32 overflow exploit	21.5
iis50_webdav	IIS 5.0 WebDav attacks	3
apache_chunked	Apache Chunked Encoding overflow exploit	31
iis_idq	IIS "IDQ" exploit	38.5
iis50_printer_overflow	IIS 5.0 printer overflow exploit	137
frontpage_fp30reg_chunked	MS Frontpage buffer overflow exploit	20.5
ddk_iis	"ddk" IIS overflow attack	32
iis_asp_overflow	IIS ".asp" overflow exploit	75
wvftpd	Wvtftp-0.9 Buffer Overflow	32
iis40_htr	IIS 4.0 .HTR Buffer Overflow	70.5
iis_w3who_overflow	IIS w3who.dll ISAPI Overflow	23
squid_ntlm_authenticate	Squid NTLM Authenticate Overflow	5.5
webstar_ftp_user	Stack overflow in OS X WebSTAR FTP server	64
blackice_pam_icq	UDP overflow targeted at ISS BlackIce program	91
windows_ssl_pct	Microsoft SSL PCT MS04-011 Overflow	9.5
windows_nsiislog_post	IIS nsiislog.dll ISAPI POST Overflow	4

(The information in this table was provided courtesy of the Skaion corporation.)

Scanning attacks are typically probing efforts with an attempt to find an unprotected port or an effort to map the network. The background attacks primarily consist of buffer overflow or known bug exploitations. It is not particularly critical to know the specifics of the attacks and scanning efforts for the purpose of this research. The critical aspect for this research is the ability to label connections either as benign or as the type of attack that they represent. If labeled attacks do not exist in the dataset under consideration, it is possible to simply run *Snort* and label as necessary. A very useful contribution to this research field would be a utility that accepts TCPdump as input and outputs vector summaries of connections with signature based alerts labeled on the flagged connections. This is a capability gap that currently exists between the networking community and the machine learning community. The technique proposed in this dissertation addresses this gap

however the computing solution is not sophisticated enough to fully overcome the aforementioned gap.

After representing this dataset as labeled connections, it was apparent that learning signature based detection schemes would be relatively trivial. Most of these detection schemes measure the same statistics created by TCPtrace, which is shown in figure 8.2.

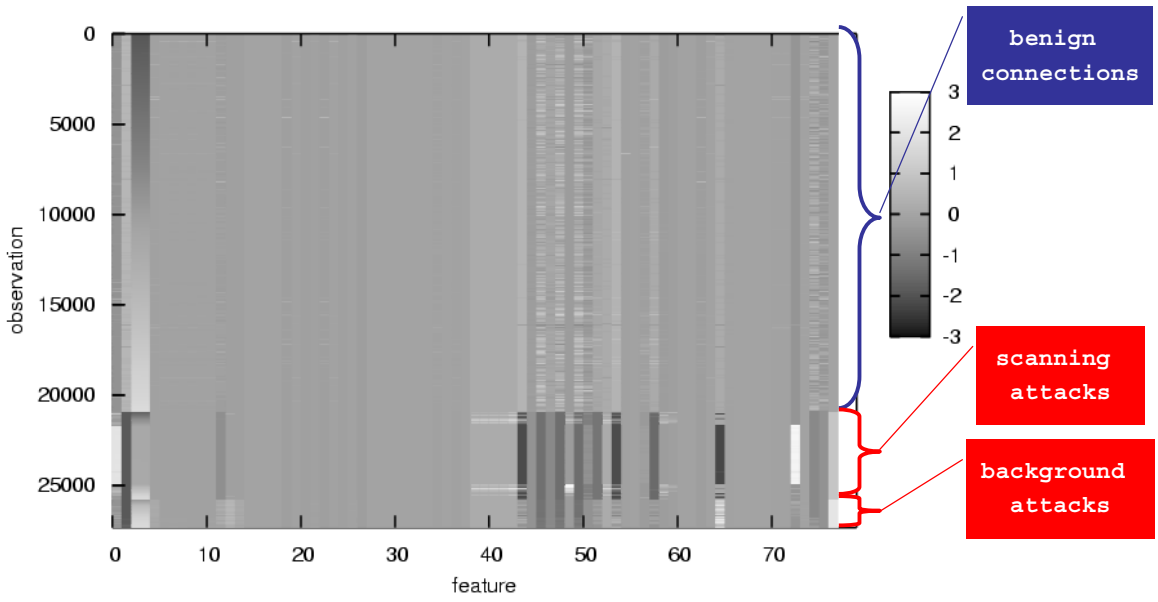

Figure 8.2: Color visualization of Skaion data

This color visualization offers much information. This plot is a color visualization plot using a new feature available with gnuplot 4.0, the PM3D plot [140]. An astute initial observation challenges the possibility of displaying 28,000 observations when this is more pixels than a computer screen contains. PM3D overcomes this problem with an elegant algorithm which fuses pixels based upon the desired color of adjacent colors, allowing for visualization of such a large dataset. The value of visualizing a dataset in this manner involves identifying obvious linear breaks in the data. The three types of connections include benign, scanning attacks, and background attacks. The last feature in this plot is the label, and therefore it should display clean linear breaks for each class. However, all other features are simply statistics from the data. Notice the clean breaks which occur in the latter half of the feature space. These features primarily involve the segment size and the window

size. The segment size involves the amount of non-header raw data transmitted in a packet, and the window size involves the ability of the receiving host to receive certain size segments. A network engineer would be interested in explaining why these are the most important features involved in identifying an attack which can be identified with signature based methods, however consider the machine learning approach to this problem. With minimal background knowledge in computer networks, the critical features used in *Snort* attacks have been identified. Exploitation of these features for the purpose of learning *Snort* signatures is now possible. What is more interesting involves the potential to identify future attacks which can be slight modifications of the attacks currently identifiable by *Snort* signatures. Machine learning applications are likely to identify slight modifications, associating these modifications with similar patterns. However, signature based detection schemes will not identify these patterns modifications. This key weakness in signature based detection, an inability to generalize due to the need to identify exact signatures, opens the door for machine learning applications.

Experimental results with this particular dataset indicated near perfection with KPLS pattern recognition, achieving an AUC of .99 or 1 with each experiment. This result is not surprising due to the obvious linear separation evident in figure 8.2.

8.3 Geospatial Datamining for Threat Templating

Much interest exists in the fields of predicting crime and threat or enemy activity. Many American cities employ data analysis experts in crime labs for the sole purpose of tracking and predicting crime patterns based upon geospatial statistics. This chapter discusses a geospatial prediction modeling application which could easily be characterized as a security classification problem.

8.3.1 Recent Work

Using geographical information to solve problems has been applied throughout history. Chawla et. al. use several historical examples in [26] to illustrate the legacy of geographical problem solving. One example included an Asiatic cholera break out

in London in 1855. Officials used a map to identify locations of the outbreak, and they identified a water pump in the middle of the locations. After turning off the water pump, the outbreak stopped. Scientists have since shown Asiatic cholera as a water borne disease. Another example included a 1909 observation that linked the healthy teeth of Colorado Springs residents to a higher than normal concentration of fluoride in drinking water for the residents. This was before scientists confirmed the importance of fluoride in fostering healthy teeth. I have personally witnessed geospatial statistics applied to common military tactics. While training at the Joint Readiness Training Center in Louisiana, a training center designed to tax light infantry units in hot, jungle environment, it is common to encounter mine strikes and mortar attacks clustered in specific areas. Intelligence officers draw equal sized circles around the locations of these mine strikes and attacks, and commanders target locations where these circles appear to meet. The circles represent the radius of a potential cache. These targeting actions typically involve ambushes and search and destroy missions looking for enemy caches. Crime research with geospatial data is a very popular field of study. Often referred to as crime mapping, applications became very wide spread across police districts in the last two decades [79]. Spatial analysis also manifests in econometrics, such as the work in [4] by Anselin.

Stephen Riese's *threatmapper* is a recent military application which seeks to find threat incident patterns by characterizing incidents as a vector of proximity measurements to specific terrain features, such as a bridge, four lane highway, large metropolitan center, etc. Stated plainly, *threatmapper* strives to find the "hotspots", areas likely to contain enemy activity, in a tactical scenario. The application is a kernel based method, considering two incidents as similar if surrounding terrain is similar however the actual distance between the two points could be quite significant. The algorithm used by Riese, known as kernel density with spatial preference, [122] follows:

$$d_g(\mathbf{x}_g) = \prod_{i=1}^{I} \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\sqrt{2\pi h_i^2}} e^{\frac{-(x_{ig}-x_{in})^2}{2h_i^2}}$$

$$\mathbf{x}_g \equiv g^{th} \text{ test instance}$$

$$n = (1, 2, ..., N) \text{ positive instances for training}$$

$$i = (1, 2, ..., I) \text{ feautures}$$

$$h_i \equiv \text{ standard deviation of the feature}$$

An initial inspection of the method reveals that *threatmapper* builds models solely based upon positive incidents, or the incidents which involved an attack. It builds models from a single class, similar to the one class SVM. However, examples from the other class are available. Excluding an entire class from a model limits the information available to a model for no known reason. Haykin indicates in [72] that one of the fundamentals in every classification or pattern recognition problems involves ensuring the inclusion of all available information. Excluding an entire class from the model violates this fundamental. Another questionable aspect involves inclusion of the standard deviation of the features rather than the more common practice in machine learning which is to assume scaled features. It is a relatively small observation, however inclusion of the standard deviation adds complexity to the model and computational complexity which is not necessary.

8.3.2 Experimental Design

The purpose of examining the *threatmapper* application was to determine whether or not the algorithm (kernel density with spatial preference) in use could be improved or replaced by another algorithm. An initial assessment of the *threatmapper* system revealed a very robust and creative geospatial method. Much of this system included sophisticated Geographic Information System (GIS) interaction with map databases and incident databases which create the eventual dataset for prediction. However, the algorithm seemed to provide room for improvement.

Five other prediction models were compared against the kernel density with spatial preference model. The experimental design divided the dataset into three groups - a training set, test set, and validation set. The training and testing set contained points from the same geographical region. The difference in the two sets simply involved a temporal change. The training set occurred during one year, and the test set occurred during the following year. This explored how well we could

predict in an area where we observed threat behavior.

Table 8.2: Experimental design for *threatmapper* experiment.

data set	number of observations	number of positive incidents (p)	number of negative incidents (b)
training	7450	420	7030
buffer zone	3155	unknown	unknown
testing	7450	200	7250
validation	7425	240	7185

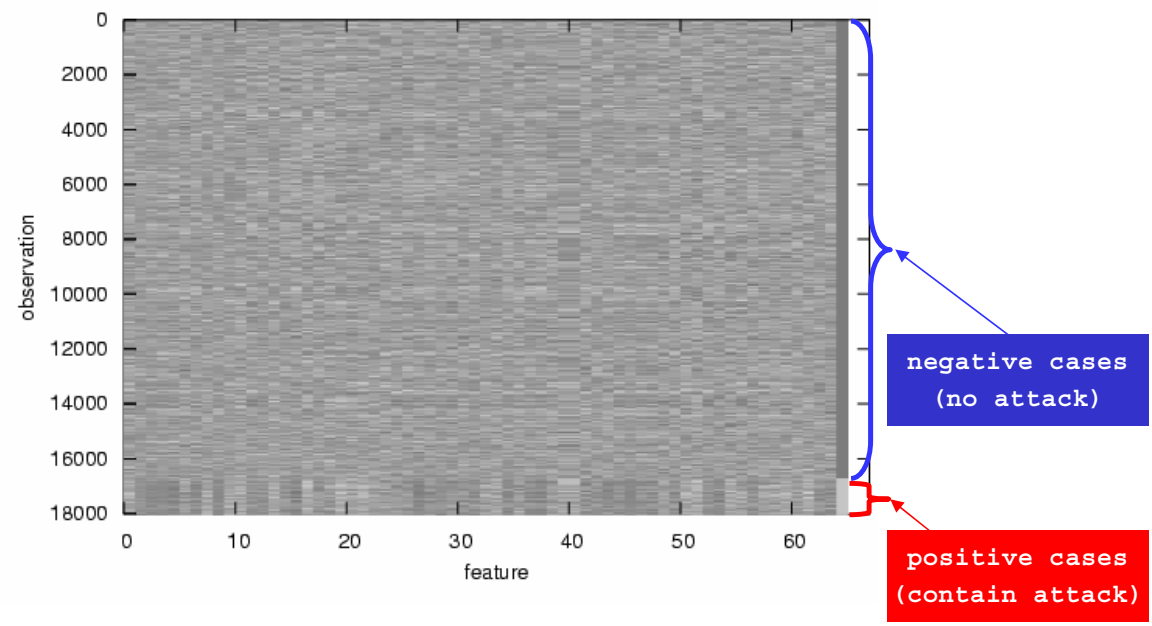

Figure 8.3: Color visualization of geospatial data

A buffer zone divided validation set from the training data and the test data. The validation set included events that occurred both during the same time as the training set and the next year. Figure 8.4 illustrates this design. Figure 8.4 represents a map, showing a cartoon depiction of the buffer zone and the relationship between the training, testing, and validation groups. Table 8.2 details the specifics of the these groups.

Figure 8.3 illustrates the data explored in this experiment. Again utilizing the power of PM3D [140] to visualize a matrix with more elements than pixels on a computer screen, this visualization clearly shows that the incident data will not be a

trivial dataset to predict. Although there seems to be a fuzzy line of separation, it is certainly not as crisp as the line of separation shown in figure 8.2. Later experiments will show that this pessimistic observation regarding the predictability of this data, strictly based thus far on data visualization, proves accurate.

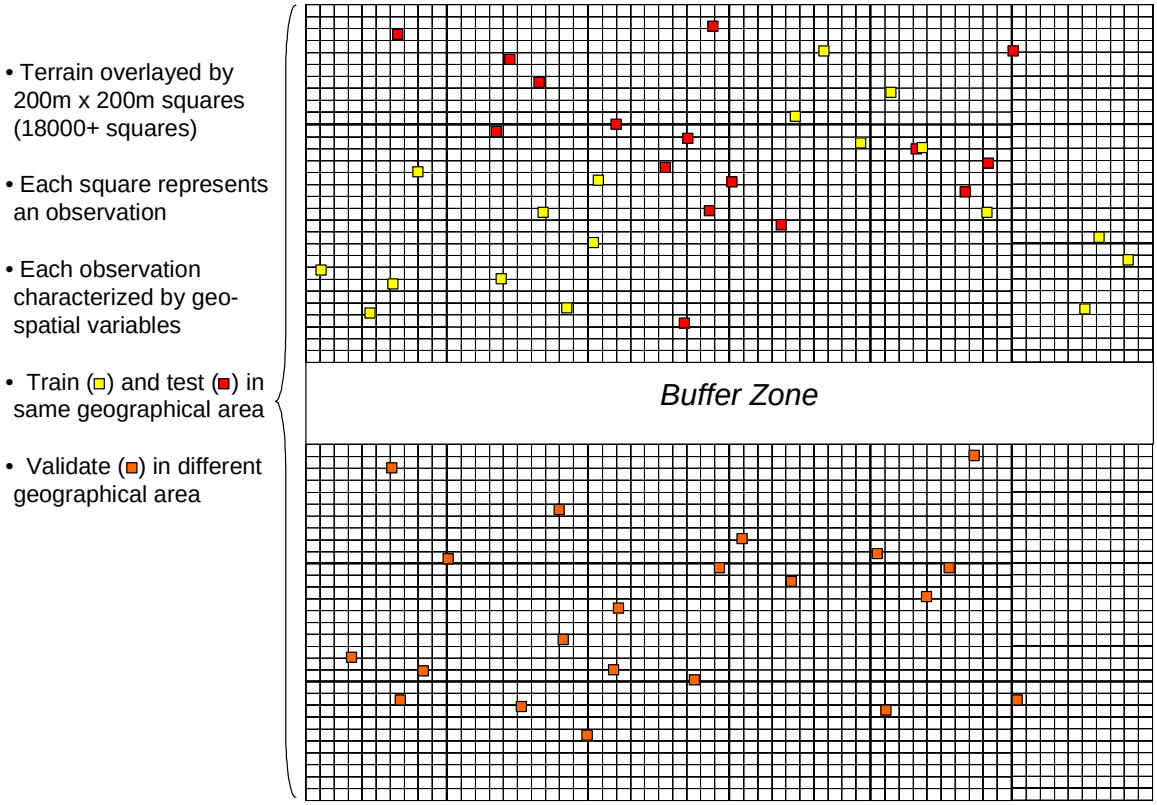

Figure 8.4: Experimental design for geospatial learning

Subtle challenges exist with geospatial data. One of these subtle challenges involves the independent and identically distributed (i.i.d.) assumptions necessary in many statistical models. This assumption is not valid with geospatial data [26]. There is inherent similarity based upon geographical proximity, which boils down to the distance between two points. Regardless of what variables a model utilizes, this underlying factor cannot be overlooked and will affect the outcome. This problem manifested itself as a significant difference in prediction performance between the test dataset and the validation dataset. Figure 8.5 illstrates the performance of six different models on the test dataset. The SVM models performed markedly better than the regression models with the test data.

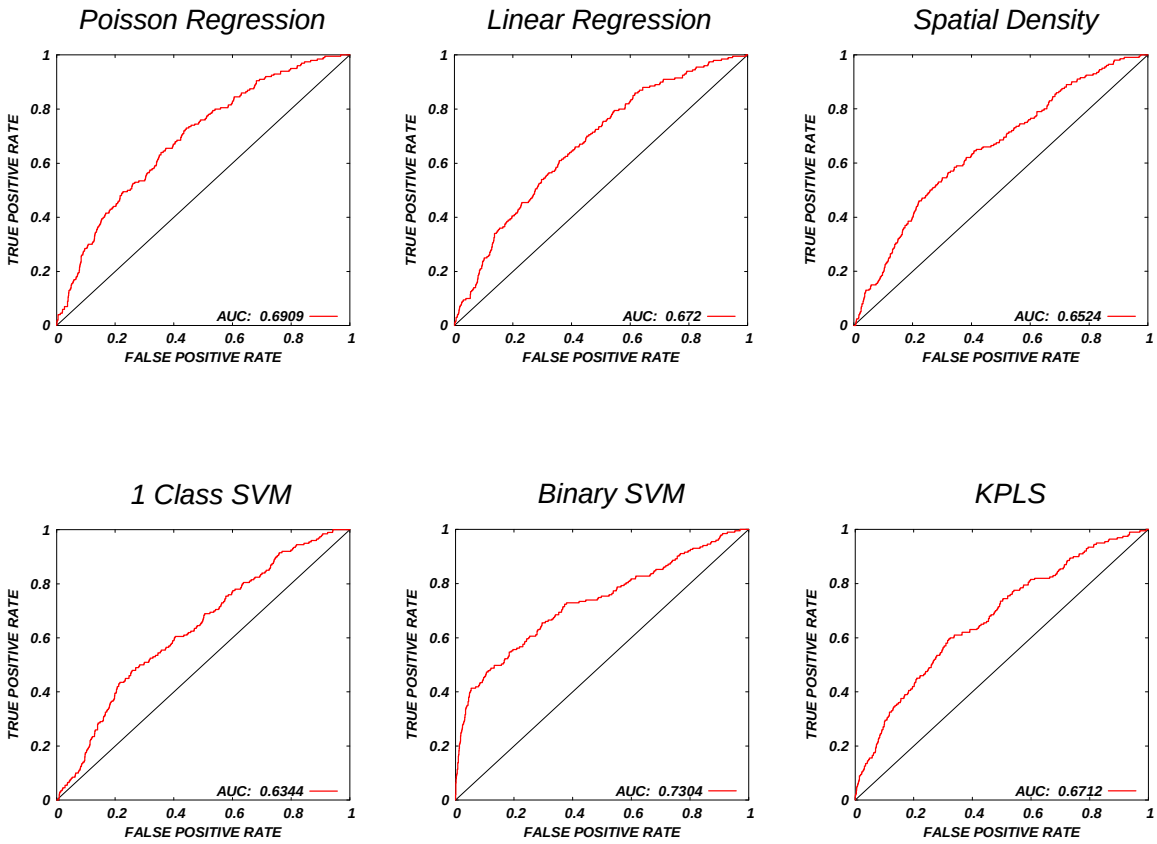

Figure 8.5: Comparison of models on the geospatial test data

Figure 8.6 illustrates the performance of six different models on the validation dataset. None of the models performed well on this data. The regression models were the better of the six however, and it is interesting to notice that the SVM models performed very poorly. The binary SVM was almost a perfectly random model.

The reason for this performance relates to the i.i.d. discussion from the previous paragraphs. Consider SVM models. SVM models are solely based upon the similarity of an evaluated point against the support vectors, or defining points, of training dataset. Regression models are built from the independent variables of the data.

Recall the one class SVM decision function where $mathbf{x}_i$ is an m dimensional vector, $i = 1, 2, ..., N$. In order to classify a new instance, $\mathbf{v}$, we would evaluate the following decision function:

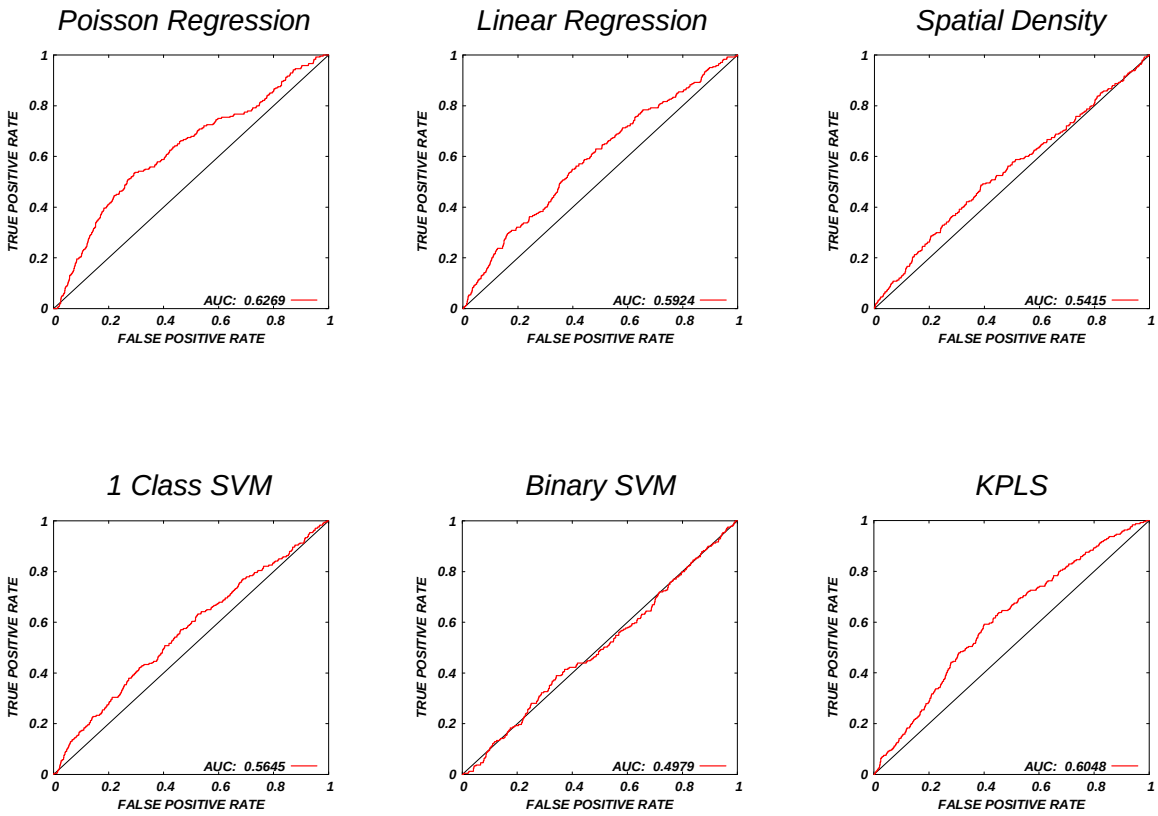

Figure 8.6: Comparison of models on the geospatial validation data

$$f(v) = \kappa(\mathbf{v}, \mathbf{v}) - 2\sum_{j} \alpha_j \kappa(\mathbf{v}, \mathbf{x}_j) + \sum_{j,k} \alpha_k \alpha_j \kappa(\mathbf{x}_k, \mathbf{x}_j) - R^2$$

The decision function of the binary SVM is somewhat simpler, however it is again defined with the α vector and kernel similarities between instances as shown by Bennett and Campbell in [8]:

$$f(v) = \sum_{i} y_i \alpha_i \kappa(\mathbf{v}, \mathbf{x}_i)$$

α is a sparse N dimensional vector, $\alpha_i \in (0, 1)$. If α_i is greater than zero, this means that the i^{th} instance is a support vector. This means that this particular instance, or vector, helps define the margin of separation in the case of the binary SVM or the enclosing hypersphere in the case of the one class SVM. SVMs create decision functions based upon specific instances from the training set. This is fundamentally different from regression models. Regression models create a function

from the independent variables, or features. This fundamental difference does help explain why the SVM models perform well in the same terrain (test dataset) but poorly on a different set of terrain (validation dataset). With geospatial data, two points that are adjacent on the terrain will have unavoidable similarities. SVM models will detect this type of similarity and always have a bias to measure geographically proximate points as similar. Many of the threat incidents occurred in the same proximity, forming clusters of incidents in the data (see figure 8.7). The poor performance of the SVMs in validation data is likely due to the models attempting to classify points based on their geographic similarity to the training data, which is a doomed effort since none of the points have geographic proximity. Regression models attempt to find the independent variables which best describe the classification function. Geographic proximity is not the critical factor in these models, thus potentially explaining why regression models performed better with the validation data.

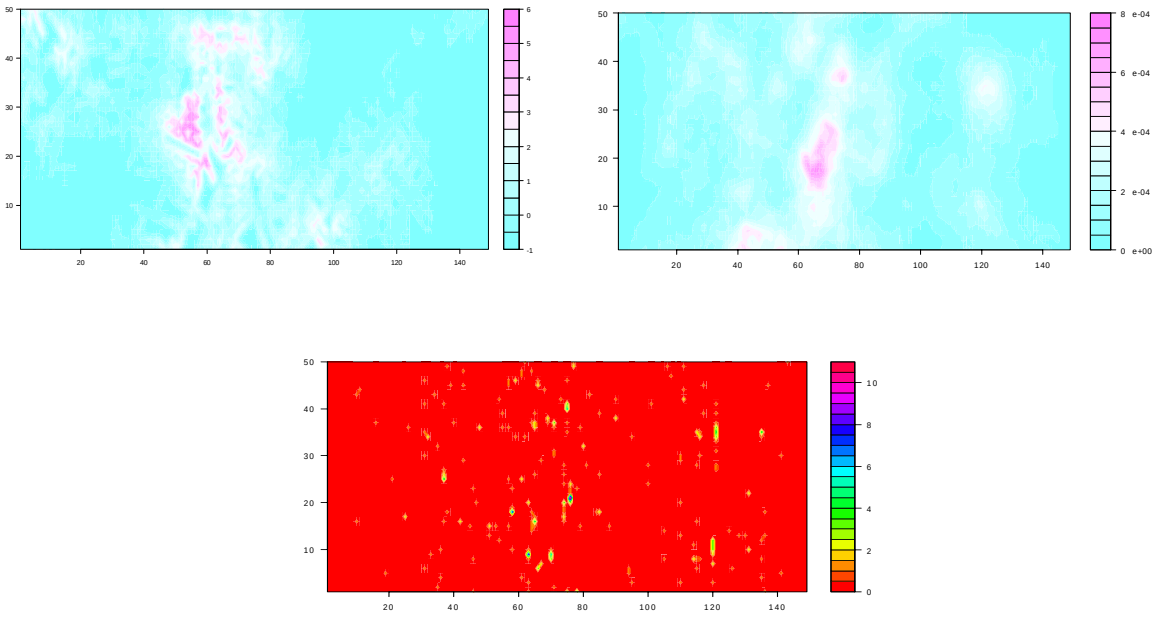

Figure 8.7: Color visualization of geospatial function

Figure 8.7 is a color map visualization of actual incident data (bottom red-colored map) and two attempted predictions of these incidents (top two maps). The purpose of this illustration is to show that many incidents did occur in clusters. Furthermore, the top two maps show how different two different prediction functions

can be with this type of data. The top left map is the kernel spatial method, and the top right is the binary SVM. The models trained on the training data (not shown) and the test data which these functions attempt to predict is the red color map on the bottom.

Prediction modeling of geospatial data has tremendous application potential. Many techniques have been created to model and visualize geospatial patterns. However, evaluation of the performance of various prediction models for geospatial data is a gap in the research. Furthermore, gaps exist involving how to create a model for one piece of terrain based upon behavior in another piece of terrain. These gaps offer opportunities for future research. The gap addressing modeling and application in separate pieces of terrain offers particular opportunity for military applications where this type of problem occurs every time a unit changes its area of responsibility.

8.4 Insights into Insurgent Response to Traffic Flow Strategies

Suicide vehicle borne improvised explosive devices (SVBIEDs) create devastating results. SVBIEDs are an intimidation tool, and the terrorists who employ these tactics hope to instill fear and uncertainty. The behavior of SVBIEDs is poorly understood and sparse. In an attempt to improve understanding of SVBIEDs and their environment, a simulation exercise with map aware non-uniform automata (MANA) ensued. The complete experiment is documented in [66]. Several of the concepts discussed in this thesis were employed in this experiment. This section will highlight the related applications from the experiment.

MANA is an agent based simulation platform. For this experiment, a VBIED released into a road network with an overall goal of detonating at an entry control point (ECP) which was located somewhere in the center of the terrain. As the VBIED progressed through the road network, the VBIED passed several TCPs. Each TCP could potentially identify and engage the VBIED, and the likelihood of this engagement was controlled by several factors which affected identification and the TCP strategy which affected VBIED exposure to TCPs.

The fundamental objective of this project involved identification and assess-

ment of key factors in determining VBIED success. Soldiers with recent experience in theatre provided information during surveys and interviews, enabling identification of the factors. Eleven important factors emerged, and these factors are shown in table 8.3. The TCP strategy deserves additional explanation. There were five different traffic control point (TCP) strategies analyzed, and each strategy involved a slightly different arrangement of TCPs on the terrain.

Table 8.3: Description of factors and measure of effectiveness

Factor Name	**Min**	**Max**	**Description**
TCP_strat	1	5	TCP strategy, numbered 1-5, representing vignettes base, filter, grid, open
CD_TCP	5	5029	Communications distance (in meters) of the traffic control point (TCP) spotters
CL_TCP	0	1824	Communications latency (in seconds) of the traffic control point (TCP) spotters
CD_ECP	5	5029	Communications distance (in meters) of the entry control point (ECP) spotters
CL_ECP	0	1824	Communications latency (in seconds) of the entry control point (ECP) spotters
Stealth	67	99	Stealth of the insurgent, measured as in index in MANA - a larger number implies better concealment
OrgRepulse	-99	61	insurgent tendency to avoid blue forces observed with organic intelligence, measured as an index in MANA - a larger value implies increased repulsion
InorgRepulse	-99	61	insurgent tendency to avoid blue forces observed with inorganic intelligence, measured as an index in MANA - a larger value implies increased repulsion
Vspeed	10	74	Speed of the insurgent after making visual contact with the entry control point (ECP), measured in kmph
b_detect	5	517	Detection range (observes an entity) in meters of blue forces
b_classify	5	517	Classification range (classifies an entity) in meters of blue forces
insurgent_y	0	1	Not a factor - primary measure of effectiveness - defined as insurgent reaching target and detonating

The max and min shown in table 8.3 spanned 33 levels in the design of experiments(DOE). The DOE utilized allowed for the analysis of 11 different factors with only 33 design points. This is a special DOE known as the nearly-orthogonal Latin hypercubes (NOLH) developed by Cioppa in [29]. Designs were readily evolved from those used in the initial study and are complimentary for comparison.

The factor descriptions and ranges are given in table 8.3. Utilizing eleven factors, 33 design points emerged. The NOLH designs require many fewer runs than more traditionally employed designs, are more efficient, and sample the interior of the factor levels hypercube [126]. After conducting 30 simulation iterations for each design point, analysis of the primary measure of effectiveness, insurgent success, followed.

Creating the 33 design points with eleven different factor values requires 341 adjustments assuming that each factor could be adjusted in one location in MANA. This was not the case. Several of the factors spanned multiple states in MANA, nearly quadrupling the total number of adjustments required for the design of experiments (DOE).

In order to support simulation validation efforts and create an agile framework for integrating the DOE, several programs were created to perform special tasks. The programs were all written in Perl and exploited MANAs underlying xml model files. Model files contain unique branch paths to every parameter (some of which were the factors), represented in a tree structure. The program built to implement the DOE followed these steps:

1. Create file vars.txt containing branch path to the m factors in the DOE.
2. Mark factors in base MANA xml file based on variables read from vars.txt..
3. Read DOE matrix and for $i = 1...n$ design points in DOE, assign $j = 1...m$ factor levels to marked variables for each design point.

This method creates n xml files, each of which runs 30 iterations. The final analysis involves aggregating 30 x n simulation output files to assess the measures of effectiveness. A key lesson learned involved building a program and file management framework robust enough to tolerate changes in the DOE and base scenario, an inevitable event. Robustness enables regeneration of results with minimal manual intervention. Another goal should also include scalability; a well constructed framework will step from one scenario and DOE to another relatively seamlessly.

8.4.1 Results and Discussion

The data generated in the MANA vignettes enabled a search for insights regarding insurgent success or failure based on factors shown in Table 8.3. These factors were identified based on feedback from participants at the Armor Officer Captains Career Course and from the literature. The data exploration involved determining significant factors and combinations of factors and building models to classify insurgents outcomes as a function of the factors.

NOLH assumes near-orthogonality of the factors. The factors displayed very little correlation, supporting this assumption as shown in figure 8.8. Examining the correlations prior to the simulation helps support this assumption; however the correlation matrix provides insight when examined with the measures of effectiveness after the simulation.

Figure 8.8 illustrates the correlation matrix on a color map, an intuitive representation which is especially useful with high dimensional datasets. Variables one through 11 in this plot respectively follow the variables listed in Table 8.3; variable twelve represents insurgent success. This plot aids analysts in discovering obvious linear relationships in the data in addition to validating data behavior. Another useful plot that enables validation and identification of obvious linear relationships is the color (grey scale) map of the actual data shown in figure 8.9.

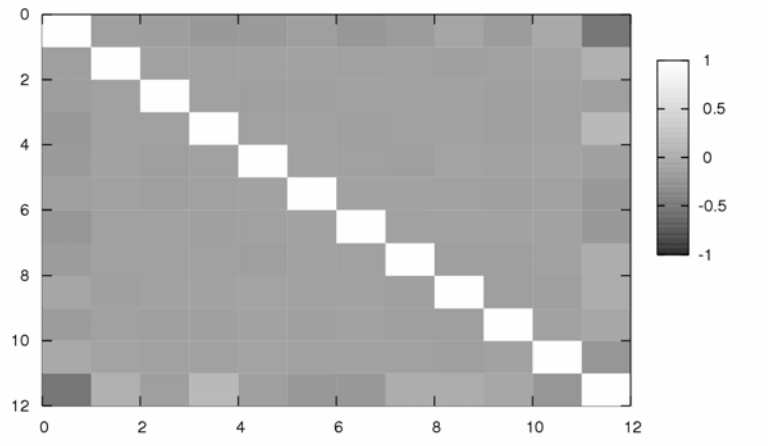

Figure 8.8: Correlation matrix of eleven factors and insurgent success

Sorted on the insurgent success measure of effectiveness (last column), the

color map provides a summary of the entire dataset. This figure illustrates relative distributions, balance of the data, and potential patterns. It provides the same type of utility for multi-dimensional data that a scatter plot provides for two dimensional data, primarily enabling validation and human understanding.

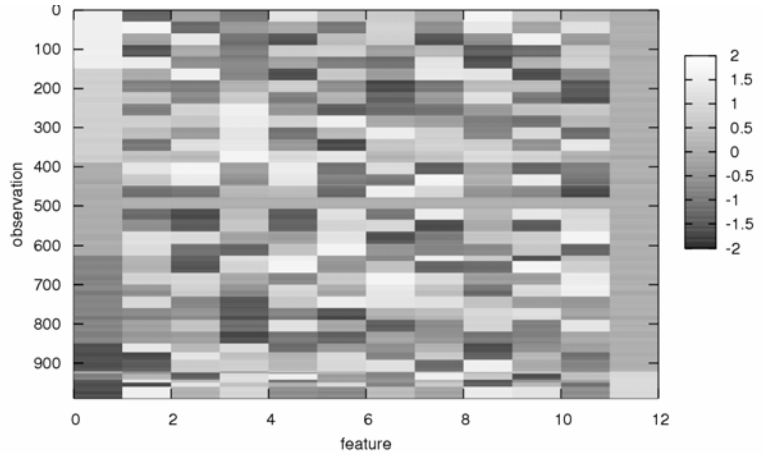

Figure 8.9: Matrix plot of data

8.4.2 Predictive modeling with MANA data farming

The data created from 990 simulation runs creates a platform for prediction modeling. Exploring simulation data for predictability potentially provides decision makers with a tool to shape their environment. The overall goal of this study aims to mitigate insurgent threat, and predictive models fill a vital gap in this effort. Since this study examined a binary measure of effectiveness (insurgent success), evaluation of the prediction models was possible with receiver-operator-characteristic (ROC) curves. It is also possible to supplement ROC curves with a decision ROC chart, a novel plot introduced in this dissertation which supplements the ROC curve by bridging the gap between the false positive rate and the decision values (see figure 8.10.

Data farming has been casually mentioned a few times and deserves explanation. Data farming involves a search of the parameter space, often involving a brute force search across many design points or a robust method for searching a large parameter space (such as in this application) with fewer design points. Barry and Koehler [7] and Horne and Meyer [77] provide much of the background for data

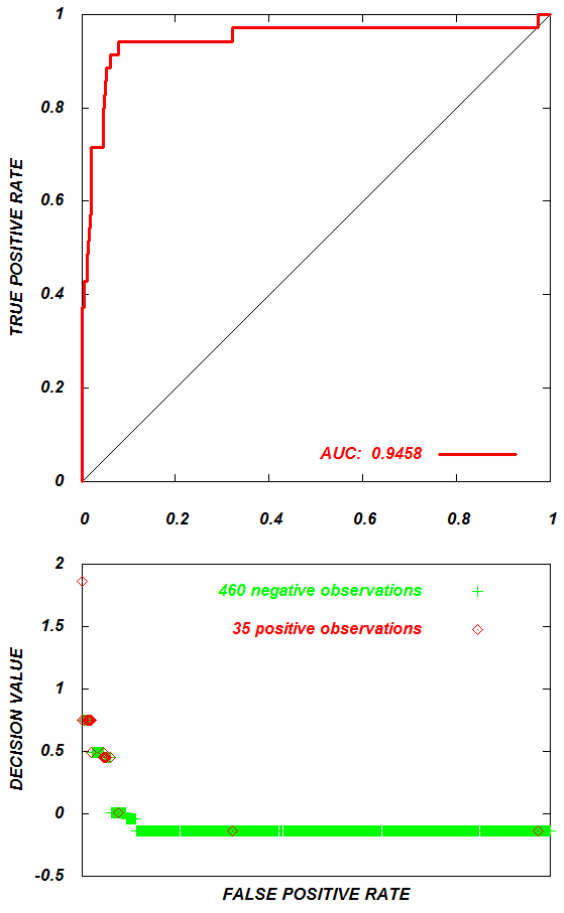

Figure 8.10: Decision ROC chart illustrating binary SVM performance for MANA data farming data

farming in their Winter Simulation Conference papers. The purpose of data farming is to detect previously unknown relationships and explore a broad range of hypotheses [77]. The strongest relationship observed in this experiment was between the TCP strategy and VBIED success. This is shown in figure 8.8.

This experiment was as much of an academic exercise as an applied effort. As an academic exercise, this experiment provides a framework for data farming with MANA and a methodology for the type of experimental design necessary to utilize machine learning and prediction modeling. It is also an applied effort, seeking to improve our understanding of a poorly understood threat. The simplest way to critique this modeling effort would be to point out that validation is not possible simply due to our lack of understanding of VBIEDs. This is undoubtedly true. However,

the methodology and framework is valid. When we do improve our understanding of this VBIED threat or realize the need for this type of modeling framework for another application, researchers can potentially capitalize on the progress and methodology set forth in this experiment.

CHAPTER 9
CONCLUSION AND FUTURE DIRECTIONS

This dissertation aims to improve the modeling of security classification problems. There are two major thrusts of the contributions. One thrust involves advances in the theory and methodology that tackles unbalanced classification. The other thrust addresses applications, illustrating techniques and methods to solve specific types of security classification problems. This conclusion includes comments on both of those major thrusts as well as some final remarks regarding future directions for this research.

9.1 Contributions to Theory and Methodology

Security classification problems, and unbalanced classification in general, are a special type of classification problem. The typical modeling and evaluation efforts used for balanced problems are often inadequate for unbalanced cases. Balanced classification modeling implicitly or explicitly makes a number of assumptions which are not always appropriate for unbalanced problems. For that reason, unbalanced classification deserves to be considered a special case by its own. The research contributions in this dissertation stem from the fact that these assumptions do not consistently hold for unbalanced classification. When these assumptions hold true, models perform adequately. However, when these assumptions do not hold true, implications and alternative methods must be considered to improve model accuracy. Unbalanced classification offers a special case when these assumptions do not hold, presenting a gap in the methods available. This dissertation has largely been an exploration of these gaps. The following itemized list includes the theoretical and methodological contributions of this work. These contributions have been associated with an assumption from balanced classification that does not hold with unbalanced classification. This provides illustration of the gaps that exist and the efforts and progress made to fill the gaps.

- *Balanced classification assumption 1: Feature reduction is possible.* This dis-

sertation clearly shows the impact of increased dimensionality (shown in sections 1.5.6, 1.6, and 2.2.3.3). More is not necessarily better when considering dimensions, or features. In its most simple form, feature reduction for balanced classification involves measuring a feature's behavior with one class, comparing that behavior with the other class, and discarding features which do not demonstrate a different behavior. There are much more advanced methods for feature reduction, many of which involve iterative model evaluation such as in [91]. Modeling in subspaces and aggregating the output of subspace models (shown in chapters 5 and 6) creates a platform to reduce dimensionality without the loss of information.

- *Balanced classification assumption 2: When aggregating the decision values from an ensemble of models, the average works best.* Bagging, an ensemble method introduced by Breiman in [18], is one of the most popular and successful ensemble methods in the literature. The aggregation method utilized is the average, and the use of the average for combining numeric model output has essentially become the assumed norm. Chapter 6 clearly illustrates a case for considering alternate aggregation methods, especially when fusing unbalanced problems. Moreover, chapter 6 shows that as the severity of imbalance increases, the value of including the min aggregator as part of the fusion method increases. An important aspect of this property requires that the fused values are ranks, eliminating the parametric influence of actual decision values. The analysis of rank distributions, pseudo ROC curves, and the behavior of these ranks when considering an imbalanced classification problem (shown in chapter 6) comprise what I consider the most valuable findings of this research.

- *Balanced classification assumption 3: Model tuning is possible through iterative training and testing.* Model validation with actual positive instances may be possible with some unbalanced classification problems, however many problems include very little or no examples of the positive class. When the latter is the case, model tuning with iterative training and testing may not be possible.

Chapter 7 introduces an approach to overcome manual tuning and validation for the gaussian kernel. The most important contributions of that chapter involve insights into the statistical properties of the gaussian kernel and the gaussian kernel matrix. Improved understanding of this kernel could lead to automated tuning methods. Chapter 7 included one heuristic for automated tuning that is based on the statistical properties presented in the chapter.

- *Balanced classification assumption 4: ROC curves and the area under the curve provide sufficient evaluation of model performance.* Evaluation of models is an open area of research. Good evaluation metrics provide model assessment that decision makers understand, evaluate without bias across a broad spectrum of model performance, and communicate complete information. ROC curves and the area under the curve are excellent performance measures for binary classification, however they are not complete. A small but important contribution of this research involves the *decision ROC chart*, a novel method (shown in chapter 3) that augments the ROC curve with a graph depicting the relationship between decision values and the false positive rate. Furthermore, the chart communicates the balance of the problem. This contribution presents a more complete picture of model performance than the ROC curve alone.

9.2 Contributions to Applications

There are two primary application domains addressed in this dissertation. Military tactical applications and computer intrusion detection applications. These domains are riddled with security considerations. These considerations motivated coining of the term "security classification problem", an environment characterized with severe or nearly complete imbalance. This is a very common environment in security scenarios.

Chapter 4 and chapter 8 address methods utilized to model military tactical classification problems and computer intrusion detection classification problems. Including these applications and describing the models utilized shares knowledge explaining how to measure these two types of scenarios. The most difficult aspect

of modeling is quite often simply getting started. Determining what and how to measure often creates unexpected challenges. Preprocessing unruly, alphanumeric data creates challenges as well.

Chapter 4 describes the text mining techniques used to measure host based intrusion detection data. This chapter also illustrated methods to measure network based intrusion detection. Much of today's network intrusion detection involves strict rule-based methods designed to detect signatures of known attacks. Utilizing existing and freely available tools, chapters 4 and 8 show how to measure network traffic and use machine learning techniques to learn from these signature based methods. The value of this work involves the potential to detect previously unseen attacks which may slightly vary from previously known signatures. Slightly modified malicious events can evade signature based detection, however machine learning methods provide a means to potentially recognize these attacks.

Chapter 8 also includes two military scenarios modeled and preprocessed to harness the predictive power of machine learning methods. One scenario involves geospatial models, and the other scenario is a simulation scenario which explores a relatively large parameter spectrum for a simulation model. Both of these scenarios illustrate the applicability of framing military scenarios, which easily extend to other physical security scenarios such as crime prevention, as security classification problems.

A crucial lesson learned from these applied cases involves computing tools (see Appendix A). The majority of the work included in this dissertation would not have been possible without the use of Perl, a high level programming language, and the UNIX computing environment. Perl enables rapid prototype development and handles messy data extremely well. Understanding how to compute in the UNIX environment provides opportunity to batch large experiments. These tools provide the freedom and power to build and run experiments seeking to illustrate theoretical results or simply provide proof of principle.

9.3 Future Directions

Tremendous opportunity for future work exists in many different areas that this research explores. Several major research domains have been touched in this dissertation, and this section will be organized in accordance with these domains.

9.3.1 Computer Intrusion Detection

As long as computer technology continues to grow, computer intrusion will always be a research rich field. This is largely due to the parallel growth of malicious efforts. The term intrusion detection has actually grown somewhat outdated and out of vogue with most of the commercial detection efforts, replaced by intrusion *prevention* systems. The argument claims that detection is too late. Prevention stops malicious activity before it starts. Stopping new malicious acts before the security community officially recognizes the signatures, typically after an exploit, is a promising direction. Part of the analysis discussed in chapter 8 directly relates to this direction, and machine learning and pattern recognition techniques offer potential to capitalize on this future research direction. Pattern recognition exists to identify similarity - perhaps not exact matches, but similar. New malicious acts often grow or mutate from existing successful malicious acts. Identification of these malicious acts could result from pattern recognition that identifies similarity.

Host-based intrusion detection also offers opportunity for future research. Many intranets that serve to facilitate communications within organizations attempt to maintain user and administrator security. The typical security measures taken involve a user ID / password and compartmentalized privileges. However, user identification could be much more sophisticated than this. Just as the Schonlau data explores techniques to measure a user's vocabulary and command behavior, measurement of other computer behaviors is possible. Measurement of keystroke behavior, mouse movement and clicking behavior, and program usage offer similar behavior matching opportunities. Profiling a user's computer behavior then creates the potential to monitor the user for extreme changes in behavior, indicating a potential breach in security or fraudulent login.

9.3.2 Tuning of the Gaussian Kernel

Improving the automation of any part of machine learning helps accomplish an overarching goal of the field - building intelligent machines. The unique statistical properties of the gaussian kernel provide opportunity to automate tuning. It is undoubtedly true that there is a range for σ that exists for every classification problem which creates a stable kernel matrix that generalizes well for the data. The gap that exists in this effort involves theoretical proof, perhaps as a bound, that relates the statistics of the kernel to the generalization of the model.

9.3.3 Ensemble Methods

Ensemble methods are a relatively new and exciting field in machine learning. One future direction that could be identified relates directly to the above subsection. This effort would pursue automation of ensembles in general. The analysis of data and identification of the best type of model and ensemble is largely a human task today. Automation of this task would be a valued contribution.

An additional potential direction which relates to chapter 6 involves finding an analytical solution for rank distributions. The rank distributions described in chapter 6 include parameters which describe the accuracy of rankings and the balance of the classes. In order to overcome not knowing the analytical solution of the distributions, simulation was utilized. An analytical solution to these distributions would contribute immensely to improve understanding of ensembles and the ranks which create ROC curves.

9.3.4 Intelligent Selection of Subspaces and Dimensionality Reduction with ICA

Chapter 5 discusses several proposed methods for intelligent subspace creation. There is no theoretical proof that shows the creation of intelligent subspaces improves performance beyond simply randomly selecting subspaces. This dissertation includes much exploratory effort which addresses intelligent subspace selection, to include several proposed methods, however underlying theoretical investigation is lacking. The contribution of this future direction would not only include improved

ensembles but could also be considered a means for dimensionality reduction for unsupervised learning.

Independent Component Analysis (ICA) as a means for feature reduction is another avenue for exploration. The application of ICA typically involves signal detection and noise reduction. ICA creates a linear transformation of data, similar to PCA. However, there is no current method to identify the information content of independent components. For every principal component, an eigen value represents the amount of information contained in the principal component (via the related loading vector or eigen vector, see section 2.2.1.1). No such measure exists for independent components. This type of measure could render ICA as a feasible feature reduction method.

9.3.5 Geospatial Prediction Modeling

Geospatial prediction modeling are often applications of security classification problems. Geospatial prediction and geospatial statistics contain subtle challenges. Correlation will always exist between two points with geographical proximity, and models which train and test on the same terrain will always contain this bias. This bias creates challenges when validating or attempting to apply models to a different piece of terrain. The application of this problem manifests often in military scenarios. Units may analyze the enemy behavior within a certain set of terrain and expect this behavior to generalize to another set of terrain when the unit moves for a future operation. Crime analysis may also encounter this challenge when analyzing similar crimes or the same criminal on two separate pieces of terrain. Furthermore, since geospatial analysis always summarizes two a two dimensional map representation, opportunity exists with visualization techniques. Geospatial prediction modeling and geospatial pattern analysis have become very popular amongst the military and crime enforcement, however it is a young field with opportunity for growth.

9.4 Concluding Remarks

The applicability of the security classification problem is extensive today, and the domains include information security, physical security, and infrastructure se-

curity. It is a ubiquitous problem that impacts our way of life. Unbalanced binary classification problems exist in medicine, the military, computer networks, and homeland security. Significant decisions must be made that boil down to a simple yes or no answer. The research in this document aimed to improve our ability to provide the correct response to important decisions.

REFERENCES

[1] Charu C. Aggarwal and Philip S. Yu. Outlier Detection for High Dimensional Data. Santa Barbara, California, 2001. Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data.

[2] Raven Alder, Jacob Babbin, Adam Doxtater, James C. Foster, Toby Kohlenberg, and Michael Rash. *Snort 2.1: Intrusion Detection*. Syngress Publishing, second edition, 2004.

[3] Kamal Ali and Michael J. Pazzani. Error Reduction through Learning Multiple Descriptions. *Machine Learning*, 24(3):173–202, 1996.

[4] Luc Anselin. *Spatial Econometrics: Methods and Models*. Kluwer Academic Publishers, 1988.

[5] Y. Alp Aslandogan and Gauri A. Mahajani. Evidence Combination in Medical Data Mining. Las Vegas, Nevada, 2004. IEEE International Conference on Information Technology, Coding and Computing.

[6] Lee J. Bain and Max Engelhardt. *Introduction to Probability and Mathematical Statistics*. Duxbury, second edition, 1991.

[7] Philip Barry and Matthew Koehler. Simulation in Context: Using Data Farming for Decision Support. In *Proceedings of the 2004 Winter Simulation Conference*, Washington, D.C., December 2004.

[8] Kristin P. Bennett and Colin Campbell. Support Vector Machines: Hype or Hallelujah. *SIGKDD Explorations*, 2(2), 2001.

[9] Kristin P. Bennett and Mark J. Embrechts. An Optimization Perspective on Partial Least Squares. In Johan Suykens and Gabor Horvath, editors, *Advances in Learning Theory: Methods, Models and Applications*, NATO Science Series III: Computer & Systems Sciences, Volume 190, pages 227–250. IOS Press Amsterdam, 2003.

[10] Dimitri P. Bertsekas and John N. Tsitsiklis. *Introduction to Probability*. Athena Scientific, 2002.

[11] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When Is "Nearest Neighbor" Meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[12] Jinbo Bi and Kristin P. Bennett. Regression Error Characteristic Curves. Washington, D.C., 2003. Proceedings of the Twentieth International Conference on Machine Learning.

[13] Ella Bingham. *Advances in Independent Component Analysis with Applications to Data Mining*. PhD thesis, Helsinki University of Technology, 2003.

[14] Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, and Johathan Tivel. Data Mining for Network Intrusion Detection: How to Get Started. MITRE Technical Report, August 2001.

[15] Piero Bonissone, Kai Goebel, and Weizhong Yan. Classifier Fusion using Triangular Norms. pages 154–163, Cagliari, Italy, June 2004. Proceedings of Multiple Classifier Systems (MCS) 2004.

[16] Joan Fisher Box. R.A. Fisher and the Design of Experiments, 1922-1926. *The American Statistician*, 34(1):1–7, February 1980.

[17] Andrew P. Bradley. The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. *Pattern Recognition*, Volume 30 (7):1145–1159, 1997.

[18] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[20] Bernard Buxton, William Langdon, and S.J. Barrett. Data Fusion by Intelligent Classifier Combination. *Measurement and Control*, 34(8):229–234, 2001.

[21] Colin Campbell and Kristin P. Bennett. A Linear Programming Approach to Novelty Detection. volume 14. Advances in Neural Information Processing Systems, 2001.

[22] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble Selection from Libraries of Models. Banff, Canada, 2004. Proceedings of the 21st International Conference on Machine Learning.

[23] Huseyin Cavusoglu, Birendra Mishra, and Srinivasan Raghunathan. Configuration of Intrusion Detection Systems. Seattle, December 2003. International Conference on Information Systems (ICIS).

[24] Chih Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. http://www.scie.ntu.edu.tw/ cjlin/libsvm, Accessed 5 September, 2004.

[25] Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning*, 46(1-3):131–159, 2002.

[26] Sanjay Chawla, Shashi Shekhar, Weili Wu, and Uygar Ozesmi. Modeling Spatial Dependencies for Mining Geospatial Data. In *Proceedings of the First SIAM International Conference on Data Mining*, Chicago, IL, April 2001.

[27] Yunqiang Chen, Xiang Zhou, and Thomas S. Huang. One-Class SVM for Learning in Image Retrieval. Thessaloniki, Greece, 2001. Proceedings of IEEE International Conference on Image Processing.

[28] Calvin S. Chu, Ivor W. Tsang, and James T. Kwok. Scaling up Support Vector Data Description by Using Core-Sets. In *Proceedings of the International Joint Conference on Neural Networks*, pages 422–430, Budapest, Hungary, July 2004.

[29] Thomas M. Cioppa. *Efficient Nearly Orthogonal and Space-filling experimental designs for High-Dimensional Complex Models*. PhD thesis, Naval Postgraduate School, 2002.

[30] Scott Coull, Joel Branch, Eric Breimer, and Boleslaw K. Szymanski. Intrusion Detection: A Bioinformatics Approach. pages 24–33, Las Vegas, Nevada, December 2003. Proceedings of the 19th Annual Computer Security Applications Conference.

[31] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel Based Learning Methods*. Cambridge University Press, 2003.

[32] Tim Crothers. *Implementing Intrusion Detection Systems*. Wiley Publishing, Inc., 2003.

[33] Jesse Davis and Mark Goadrich. The Relationship Between Precision-Recall and ROC curves. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 233–240. ACM Press, 2006.

[34] James M. DeLeo. Measuring Classifier Intelligence. pages 694–699. Proceedings of the 2002 PerMIS Workshop, 2001.

[35] James M. DeLeo and Gregory Campbell. Fundamentals of Fuzzy Receiver Operating Characteristic (ROC) Functions. pages 543–548. Proceedings of the 21st Symposium Interface, 1989.

[36] James M. DeLeo and Gregory Campbell. The Fuzzy Receiver Operating Characteristic Function and Medical Decisions with Uncertainty. pages 694–699. Proceedings of the First International Symposium on Uncertainty Modeling and Analysis, 1991.

[37] Dorothy Denning and Peter J. Denning, editors. *Internet Besieged*. ACM Press, 1998.

[38] Dorothy E. Denning. An Intrusion Detection Model. pages 222–232, Oakland, CA, 2003. Proceedings of the IEEE Symposium on Security and Privacy.

[39] Lori E. Dodd and Margaret S. Pepe. Partial AUC Estimation and Regression. *University of Washington Biostatistics Working Paper Series*, Working Paper 181, January 13 2003.

[40] Robert P. W. Duin and David M. J. Tax. Classifier Conditional Posterior Probabilities. In *SSPR '98/SPR '98: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, 1998.

[41] William DuMouchel, Wen Hua Ju, Alan F. Karr, Matthius Schonlau, Martin Theus, and Yehuda Vardi. Computer Intrusion: Detecting Masquerades. *Statistical Science*, 16(1):1–17, 2001.

[42] William DuMouchel and Matthius Schonlau. A Fast Computer Intrusion Detection Algorithm Based on Hypothesis Testing of Command Transition Probabilities. pages 189–193. The Fourth International Conference of Knowledge Discovery and Data Mining, August 1998.

[43] William DuMouchel and Matthius Schonlau. A Comparison of Test Statistics for Computer Intrusion Detection Based on Principal Components Regression of Transition Probabilities. pages 404–413. Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics, 1999.

[44] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[45] James P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press, Inc., 2003.

[46] Mark J. Embrechts. *Analyze*^{TM}, Version 6.86, Rensselaer Polytechnic Institute. http://www.drugmining.com, Accessed 4 June, 2004.

[47] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, and Pang-Ning Tan. Detection of Novel Network Attacks Using Data Mining. In *Proceedings of the ICDM Workshop on Datamining for Computer Security (DMSEC)*. Melbourne, Florida, 2003.

[48] Paul F. Evangelista, Piero Bonissone, Mark J. Embrechts, and Boleslaw K. Szymanski. Fuzzy ROC Curves for the One Class SVM: Application to Intrusion Detection. In *Proceedings of the International Joint Conference on Neural Networks*, Montreal, Canada, August 2005.

[49] Paul F. Evangelista, Piero Bonissone, Mark J. Embrechts, and Boleslaw K. Szymanski. Unsupervised Fuzzy Ensembles and Their Use in Intrusion Detection. In *Proceedings of the European Symposium on Artificial Neural Networks*, Bruges, Belgium, April 2005.

[50] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Computer Intrusion Detection Through Predictive Models. pages 489–494, St. Louis, Missouri, November 2004. Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining and Complex Systems.

[51] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Data Fusion for Outlier Detection through Pseudo ROC Curves and Rank Distributions. In *Proceedings of the International Joint Conference on Neural Networks*, Vancouver, Canada, July 2006.

[52] Paul F. Evangelista, Mark J. Embrechts, and Boleslaw K. Szymanski. Taming the Curse of Dimensionality in Kernels and Novelty Detection. In Ajith Abraham, Bernard de Baets, Mario Köppen, and Bertam Nickolay, editors, *Applied Soft Computing Technologies: The Challenge of Complexity*, pages 431–444. Springer Verlag, 2006.

[53] Jerome Fan, Suneel Upadhye, and Andrew Worster. Understanding Receiver Operating Characteristic (ROC) Curves. *Canadian Journal of Emergency Medicine*, 8(1):19–20, 2005.

[54] Tom Fawcett. Using Rule Sets to Maximize ROC Performance. San Jose, CA, 2001. IEEE International Conference on Data Mining.

[55] Tom Fawcett. ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Palo Alto, CA, 2003. Technical Report HPL-2003-4, Hewlett Packard.

[56] Tom Fawcett and Foster Provost. Combining Data Mining and Machine Learning for Effective User Profiling. pages 55–62. Proceedings on the Second International Conference on Knowledge Discovery and Data Mining, 1996.

[57] Tom Fawcett and Foster Provost. Robust Classification for Imprecise Environments. *Machine Learning Journal*, 42(3):203–231, 2001.

[58] Ronald Aylmer Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[59] Ronald Aylmer Fisher. *Collected Papers of R.A. Fisher (5 vols.)*. Adelaide: University of Adelaide, 1974.

[60] Jeremy Frank. Artificial Intelligence and Intrusion Detection: Current and Future Directions. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, MD, 1994.

[61] Ana L.N. Fred and Anil K. Jain. Robust Data Clustering. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 128–133, Madison, Wisconsin, 2003.

[62] D. Frossyniotis, A. Likas, and A. Stafylopatis. A Clustering Method Based on Boosting. *Pattern Recognition Letters*, 25:641–654, 2004.

[63] Miguel Angel Garcia and Domenec Puig. Robust Aggregation of Expert Opinions Based on Conflict Analysis and Resolution. In R. Conejo, M. Urretavizcaya, and J.L. Perez de la Cruz, editors, *Lecture Notes in Artificial Intelligence 3040, Current Topics in Artificial Intelligence, 10th CAEPIA 2003 and 5th TTIA 2003, Revised Selected Papers*, pages 488–497. Springer-Verlag, 2004.

[64] Giorgio Giacinto, Fabio Roli, and Luca Didaci. Fusion of Multiple Classifiers for Intrusion Detection in Computer Networks. *Pattern Recognition Letters*, 24(12):1795–1803, 2003.

[65] Andrew G. Glen, Lawrence M. Leemis, and John H. Drew. Computing the Distribution of the Product of Two Continuous Random Variables. *Computational Statistics and Data Analysis*, 44(3):451–464, 2004.

[66] Greg Griffin, Paul F. Evangelista, Simon R. Goerger, Niki C. Goerger, and Paul W. Richmond. Insights into Insurgent Decisioning and Response to Traffic Flow Strategies. Orlando, FL, September 2006. 2006 Fall Simulation Interoperability Workshop.

[67] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti Zadeh, editors. *Feature Extraction, Foundations, and Applications*. Springer, 2006.

[68] J.W. Haines, R.P. Lippmann, D.J. Fried, M.A. Zissman, E. Tran, and S.B. Boswell. 1999 DARPA Intrusion Detection Evaluation: Design and Procedures. Cambridge, Massachusetts, 2001. Massachusetts Institute of Technology Lincoln Laboratory, Technical Report 1062.

[69] Frederic M. Ham and Ivica Kostanic. *Principles of Neurocomputing for Science and Engineering*. John Wiley and Sons, Inc., 2001.

[70] Thomas M. Hamill. Interpretation of Rank Histograms for Verifying Ensemble Forecasts. *Monthly Weather Review*, 129(3):550–560, 2000.

[71] James A. Hanley and Barbara J. McNeil. The Meaning and Use of the Area Under the Receiver Operating Characteristic Curve. *Radiology*, 143(1): 29–36, 1982.

[72] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 1999.

[73] Katherine Heller, Krysta Svore, Angelos Keromytis, and Salvatore Stolfo. One Class Support Vector Machines for Detecting Anomalous Windows Registry Accesses. In *Proceedings of the ICDM Workshop on Datamining for Computer Security (DMSEC)*. Melbourne, Florida, 2003.

[74] Tin Kam Ho. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[75] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision Combination in Multiple Classifier Systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:66–75, 1994.

[76] Alexander Hofmann, Timo Horeis, and Bernhard Sick. Feature Selection for Intrusion Detection: An Evolutionary Wrapper Approach. Budapest, Hungary, July 2004. International Joint Conference on Neural Networks.

[77] Gary E. Horne and Ted E. Meyer. Data Farming: Discovering Surprise. In *Proceedings of the 2004 Winter Simulation Conference*, Washington, D.C., December 2004.

[78] Harold Hotelling. Relations Between Two Sets of Variates. *Biometrika*, 28 (3/4):321–377, 1937.

[79] Tianming Hu and Sam Yuan Sung. Spatial Similarity Measures in Location Prediction. *Journal of Geographic Information and Decision Analysis*, 7(2): 93–104, 2003.

[80] Aapo Hyvarinen. Survey on Independent Component Analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[81] Aapo Hyvarinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis*. John Wiley and Sons, Inc., 2001.

[82] Sushil Jajodia, Peng Ning, and X. Sean Wang. *Intrusion Detection in Distributed Systems: An Abstraction Based Approach*. Kluwer Academic Publishers, 2004.

[83] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1997.

[84] Yulei Jiang, Charles E. Metz, and Robert M. Nishikawa. A Receiver Operating Characteristic Partial Area Index for Highly Sensitive Diagnostic Tests. *Radiology*, 201:745–750, 1996.

[85] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, fifth edition, 2002.

[86] Stephen L. Johnston. The Aircraft Warning Service, Hawaii and The Signal Company, Aircraft Warning, Hawaii. *Aerospace and Electronic Systems Magazine, IEEE*, 6(12):3–7, December 1991.

[87] S. Sathiya Keerthi. Efficient Tuning of SVM Hyperparameters Using Radius/Margin Bound and Iterative Algorithms. *IEEE Transactions on Neural Networks*, 13(5):1225–1229, September 2002.

[88] S. Sathiya Keerthi and Chih-Jen Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15:1667–1689, 2003.

[89] M.G. Kendall. *Rank Correlation Methods*. Charles Griffin, London, 1948.

[90] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards Parameter-Free Data Mining. Seattle, WA, August 2004. Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[91] Robert H. Kewley, Mark J. Embrechts, and Curt Breneman. Data Strip Mining for the Virtual Design of Pharmaceuticals with Neural Networks. *IEEE Transactions on Neural Networks*, 11(3):668–679, May 2000.

[92] Ron Kohavi and Foster Provost. Glossary of Terms. *Machine Learning*, 30 (2-3):271–274, February 1998.

[93] Mario Köppen. The Curse of Dimensionality. (held on the internet), September 4-18 2000. 5th Online World Conference on Soft Computing in Industrial Applications (WSC5).

[94] Christopher Krügel, Thomas Toth, and Engin Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. In *SAC '02: Proceedings of the 2002 ACM symposium on Applied computing*, pages 201–208, New York, NY, USA, 2002. ACM Press.

[95] Ludmila I. Kuncheva. A Theoretical Study on Six Classifier Fusion Strategies. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):729–741, February 2003.

[96] Ludmila I. Kuncheva. 'Fuzzy' vs. 'Non-fuzzy' in Combining Classifiers Designed by Boosting. *IEEE Transactions on Fuzzy Systems*, 11(3):729–741, 2003.

[97] Ludmila I. Kuncheva. That Elusive Diversity in Classifier Ensembles. Mallorca, Spain, 2003. Proceedings of 1st Iberian Conference on Pattern Recognition and Image Analysis.

[98] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley and Sons, Inc., 2004.

[99] Ludmila I. Kuncheva and C.J. Whitaker. Measures of Diversity in Classifier Ensembles. *Machine Learning*, 51:181–207, 2003.

[100] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection. In *Proceedings of Third SIAM Conference on Data Mining*, San Francisco, May 2003.

[101] Wenke Lee and Salvatore J. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):227–261, 2000.

[102] Erich L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks*. Holden-Day, Inc., San Francisco, CA, 1975.

[103] F.W. Loomis. Problems of Air Defense: Final Report of Project Charles. MIT Report, August 1951.

[104] Pete Loshin. *TCP/IP: Clearly Explained*. Morgan Kaufman, third edition, 1999.

[105] Junshui Ma and Simon Perkins. Time-series Novelty Detection Using One-class Support Vector Machines. Portland, Oregon, July 2003. International Joint Conference on Neural Networks.

[106] Larry M. Manevitz and Malik Yousef. One-Class SVMs for Document Classification. *Journal of Machine Learning Research 2*, 2:139–154, 2001.

[107] H.B. Mann and D.R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[108] David J. Marchette. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. Springer-Verlag, 2001.

[109] S.J. Mason and N.E. Graham. Areas Beneath the Relative Operating Characteristics (ROC) and relative operating levels (ROC) curves: Statistical Significance and Interpretation. *Quarterly Journal of the Royal Meteorological Society*, 128:2145–2166, 2002.

[110] Roy A. Maxion. Masquerade Detection Using Enriched Command Lines. San Francisco, CA, June 2003. International Conference on Dependable Systems and Networks.

[111] Roy A. Maxion and Tahlia N. Townsend. Masquerade Detection Using Truncated Command Lines. Washington, D.C., June 2002. International Conference on Dependable Systems and Networks.

[112] Donna Katzman McClish. Analyzing a Portion of the ROC Curve. *Medical Decision Making*, 9:190–195, 1989.

[113] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., 1994.

[114] Thomas W. Miller. *Data and Text Mining*. Prentice Hall, 2005.

[115] F. Robert Naka and William W. Ward. Distant Early Warning Line Radars: The Quest for Automatic Signal Detection. *Lincoln Laboratory Journal*, 12 (2):181–204, 2000.

[116] Shawn D. Ostermann. TCPtrace, Ohio University. http://jarok.cs.ohiou.edu/software/tcptrace/index.html, Accessed 28 March 2005.

[117] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace Clustering for High Dimensional Data: A Review. *SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2004.

[118] Roger G. Petersen. *Design and Analysis of Experiments*. Marcell Dekker, Inc., 1985.

[119] John C. Platt. Probabilities for Support Vector Machines. In Alexander J. Smola, Peter Bartlett, Bernhard Schölkopf, and Dale Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[120] J.R. Quinlan. Bagging, Boosting, and C4.5. pages 97–123. Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1996.

[121] Alain Rakotomamonjy. Optimizing Area Under ROC Curve with SVMs. In *Proceedings of European Conference on Artificial Intelligence Workshop on ROC Curve and Artificial Intelligence*, Valencia, Spain, 2004.

[122] Stephen R. Riese. Templating an Adaptive Threat: Spatial Forecasting in Operations Enduring Freedom and Iraqi Freedom. *Engineer: The Professional Bulletin for Army Engineers*, pages 42–43, January-March 2006.

[123] Vijay K. Rohatgi and A.K.Md. Ehsanes Saleh. *An Introduction to Probability and Statistics*. Wiley, second edition, 2001.

[124] Roman Rosipal and Leonard J. Trejo. Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space. *Journal of Machine Learning Research*, 2:97–123, 2001.

[125] Sam Roweis. One Microphone Source Separation. pages 793–799, Denver, Colorado, 2000. Neural Information Processing Systems 13 (NIPS'00).

[126] Susan M. Sanchez and Thomas W. Lucas. Exploring the World of Agent-based Simulations: Simple Models, Complex Analyses. In *Proceedings of the 2002 Winter Simulation Conference*, Piscataway, New Jersey, 2002.

[127] Bernhard Schölkopf, John C. Platt, John Shawe Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the Support of a High Dimensional Distribution. *Neural Computation*, 13:1443–1471, 2001.

[128] Matthius Schonlau and Martin Theus. Intrusion Detection Based on Structural Zeroes. *Statistical Computing and Graphics Newsletter*, 9(1): 12–17, 1998.

[129] Matthius Schonlau and Martin Theus. Detecting Masquerades in Intrusion Detection Based on Unpopular Commands. *Information Processing Letters*, 76(1-2):33–38, 2000.

[130] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[131] Ingo Steinwart, Don Hush, and Clint Scovel. A Classification Framework for Anomaly Detection. *Journal of Machine Learning Research*, 6:211–232, March 2005.

[132] Salvatore Stolfo and Ke Wang. One Class Training for Masquerade Detection. Florida, 19 November 2003. 3rd IEEE Conference Data Mining Workshop on Data Mining for Computer Security.

[133] Alexander Strehl and Joydeep Ghosh. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3:583–617, December 2002.

[134] John A. Swets, Robyn M. Dawes, and John Monahan. Better Decisions through Science. *Scientific American*, 284(4):83–87, October 2001.

[135] Boleslaw K. Szymanski and Yongqiang Zhang. Recursive Data Mining for Masquerade Detection and Author Identification. West Point, NY, 9-11 June 2004. 3rd Annual IEEE Information Assurance Workshop.

[136] David M.J. Tax. *One-class Classification*. PhD thesis, Delft University of Technology, 2001.

[137] David M.J. Tax and Robert P.W. Duin. Support Vector Domain Description. *Pattern Recognition Letters*, 20:1191–1199, 1999.

[138] David M.J. Tax and Robert P.W. Duin. Support Vector Data Description. *Machine Learning*, 54:45–66, 2004.

[139] J.D. Tubbs and W.O. Alltop. Measures of Confidence Associated with Combining Classification Results. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:690–692, 1991.

[140] unknown. *gnuplot*, Version 4.0. http://www.gnuplot.org, Accessed 1 August, 2006.

[141] Runar Unnthorsson, Thomas Philip Runarsson, and Magnus Thor Jonsson. Model Selection in One-Class ν-SVMs using RBF Kernels. In *Proceedings of 16th International Congress and Exhibition on Condition Monitoring And Diagnostic Engineering Management (COMADEM 2003)*, Sweden, 2003.

[142] Ke Wang and Salvatore Stolfo. One Class Training for Masquerade Detection. In *Proceedings of the ICDM Workshop on Datamining for Computer Security (DMSEC)*. Melbourne, Florida, 2003.

[143] Ke Wang and Salvatore J. Stolfo. Anomalous Payload-based Network Intrusion Detection. In *Proceedings of Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004*, pages 203–222, Sophia Antipolis, France, 2004.

[144] Geoffrey I. Webb and Zijan Zheng. Multi-Strategy Ensemble Learning: Reducing Error by Combining Ensemble Learning Techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16(8):980–991, 2004.

[145] Frank Wilcoxon. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[146] Herman Wold. Estimation of Principal Components and Related Models by Iterative Least Squares. In P. R. Krishnaiah, editor, *Multivariate Analysis*, pages 391–420. Academic Press, NY, 1966.

[147] Herman Wold. Path Models with Latent Variables. In H.M. Ballock, editor, *Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building*, pages 307–357. Academic Press, NY, 1975.

[148] Svante Wold. Personal Memories of the Early PLS Development. *Chemometrics and Intelligent Laboratory Systems*, 58:83–84, 2001.

[149] Svante Wold, Michael Sjöström, and Lennart Erikson. PLS Regression: A Basic Tool of Chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

[150] Jiong Yang, Wei Wang, Haixun Wang, and Philip Yu. δ-clusters: Capturing Subspace Correlation in a Large Data Set. pages 517–528. 18th International Conference on Data Engineering, 2004.

[151] Nong Ye and Qiang Chen. An Anomaly Detection Technique Based on a Chi-Square Statistic for Detecting Intrusions into Information Systems. *Quality and Reliability Engineering International*, 17:105–112, 2001.

[152] Dong D. Zhang, Xia-Hua Zhou, Daniel H. Freeman Jr., and Jean L. Freeman. A Non-parametric Method for the Comparison of Partial Areas Under ROC Curves and Its Application to Large Health Care Data Sets. *Statistics in Medicine*, 21:701–715, 2002.

[153] Jian Zhang, Yiming Yang, and Jaime Carbonell. New Event Detection with Nearest Neighbor, Support Vector Machines, and Kernel Regression. Pittsburgh, PA, April 2004. Technical Report CMU-CS-04-118, Carnegie Mellon University.

APPENDIX A
Comments on Computing

A.1 Some of the Technology Utilized

There are several computing techniques and programs that enabled the research contained in this thesis. The numerical experiments detailed in the previous chapters are a result of programs written by others, programs written by me, and the availability of some useful computing platforms. Some of the techniques and programs utilized are worth discussing. The purpose of this discussion is to share tools that worked well and comment on the context of their use.

Most of the numerical computing was completed in a $UNIX$ environment. Some of this occurred on entirely $UNIX$ based systems, however much of the development and analysis occurred on my PC with a program that replicates a $UNIX$ environment, *cygwin*. *cygwin* enabled the development of scripts that were eventually transferred to a remote $UNIX$ machine which then ran the necessary number of experimental iterations. The remote computing program utilized was *SSH Secure Shell*, a useful application which includes both a remote terminal window and a remote ftp window.

A.2 The Computing Processes

Figure A.1 represents the computing and data processing for the security problem applied to either host based or network based computer intrusion detection. This data flow assumes subspace modeling is preferred. Each of these processes represents computing that often consists of Perl programs that I wrote to analyze and manipulate data and state of the art prediction modeling software written by others. A brief description of each of these processes follows.

- *1.0 Capture Raw Data.* This is the mechanism that captures either user commands or network packets depending upon the type of activity. This process

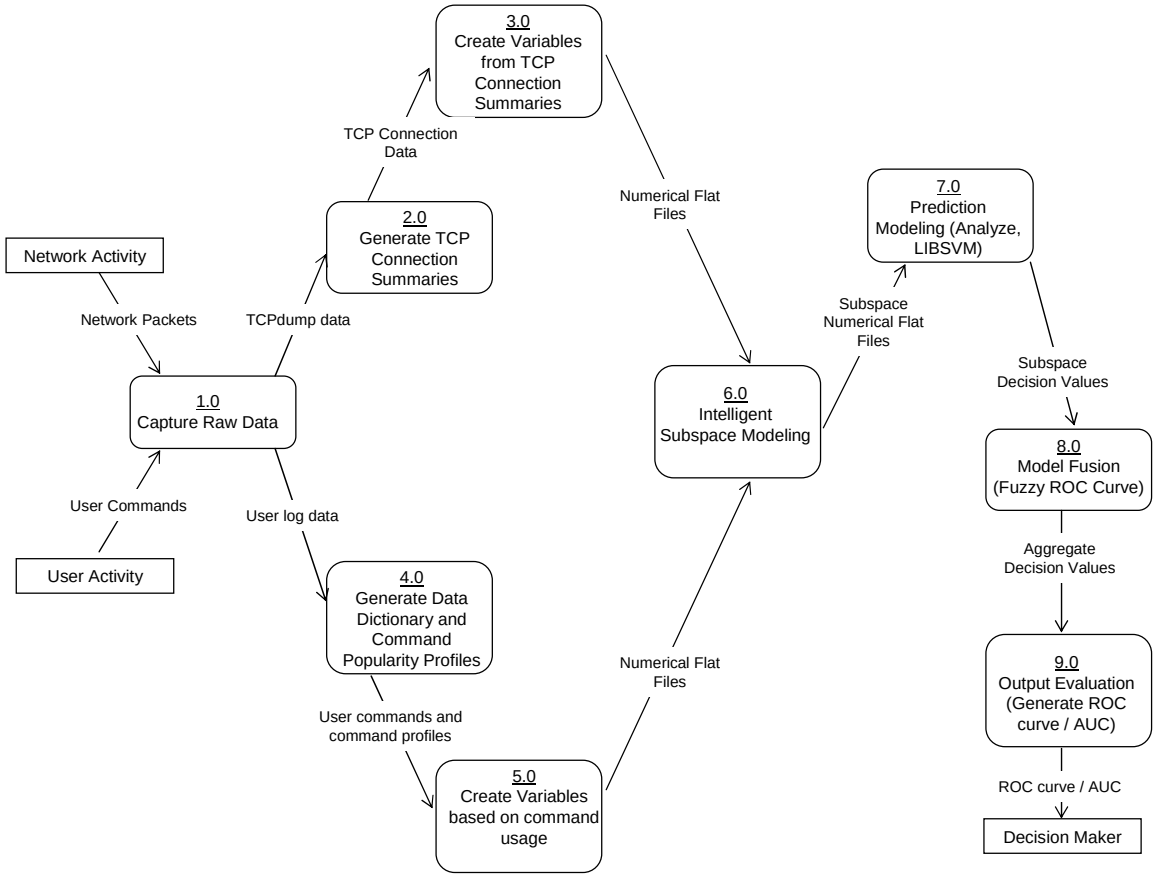

Figure A.1: Data flow diagram (DFD) of security classification applied to computer intrusion detection

will require the greatest disk storage of any process due to the raw and uncompressed nature of captured data.

The following processes apply only to network based intrusion detection.

- *2.0 Generate TCP Connection Summaries.* TCPtrace generates the connection summaries from the TCPdump data. This process can reduce the size of the data down to one tenth and represents a significant abstraction of information.

- *3.0 Create Variables from TCP Connection Summaries.* I wrote several Perl programs to parse the TCPtrace output into a numerical flat file format which is compatible with either LIBSVM or *Analyze*^{TM}. This process again reduces the size of the data to about one tenth of the size of the TCPtrace connection

summaries.

The following processes apply only to host based intrusion detection.

- *4.0 Generate Data Dictionary and Command Popularity Profiles.* User log data will consist of repetitive commands and activities which reflect the behavior of a user. In order to create statistics such as frequencies and average command popularity, a data dictionary and command popularity profile must be created. These will be created for the entire population and each user.

- *5.0 Create Variables Based on Command Usage.* User activity must be divided into sessions. Each session can be measured for consistency of activity based upon the data dictionary and command popularity profiles.

- *6.0 Intelligent Subspace Modeling.* If the variables that represent user or network activity are many (which is usually the case, especially with network data) and the models will be unsupervised, subspace modeling is preferred. Intelligent subspace modeling could include a genetic algorithm which seeks to optimize a fitness function serving as a metric for good subspaces or an analytical solution based upon the behavior of the variables.

- *7.0 Prediction Modeling.* $Analyze^{TM}$ serves as the primary software package for multivariate statistical analysis and supervised prediction modeling [46], and LIBSVM provides modeling with the one-class SVM [24]. These software packages provide a soft decision value ($\mathbb{R}^1$) which represents the model's degree of confidence towards one class or the other.

- *8.0 Model Fusion.* The Fuzzy ROC curve is the model fusion technique explained in this proposal, however many different fusion methods exist. The fusion method simply must aggregate the subspace decision values and create one decision value for ever observation.

- *9.0 Output Evaluation.* The output evaluation for binary classification will consist of ROC curve and AUC analysis. If it is not possible to create ROC curves, the decision maker must select a threshold value for classification.

Feedback of classified instances can serve as a tuning mechanism for the threshold value.

APPENDIX B
COMBINING DATA MINING TECHNIQUES WITH SCHONLAU'S DATA

B.1 Determining Optimal Variable Combinations for K-PLS Prediction

The analysis of the text mining variables described in chapter 4 provides insight into the relationship between the variables and the predictive power of these variables. Exploring fundamental properties of the variables is an important first step.

The $Analyze^{TM}$ software package is the primary vehicle for analysis from this point forward for the text mining variables. Previous analysis and preprocessing utilized Perl programs and some simple spreadsheet calculations, however now that all of the variables have been produced and represented in the MetaNeural [46] format required by $Analyze^{TM}$, the predictive modeling power of $Analyze^{TM}$ can be harnessed.

The choice of learning machines is often complicated and data dependent. This analysis utilizes one learning machine for the sake of consistency - Rosipal's kernel partial least squares [124]. This learning machine provided good results and remaining with one learning machine provides a consistent measure of comparison between preprocessing techniques and combinations of variables.

Our original set of variables did not contain variable 1 (new) or variable 8 (user). Therefore, our original model contained seven variables. The most obvious method for prediction modeling given the set of variables described is to simply represent each tuple of data with a combination of these seven variables. Let us call this the seven feature model. This is the initial approach. The file to be processed by the learning machine contained 5000 tuples of data, each containing 7 variables, the outcome (-1 for non intrusion, 1 for intrusion), and an identification number. The identification number is a six digit number that uniquely identifies each tuple. The

first three digits represent the user, and these digits range from 101 to 150. We use three digits to eliminate the confusion of any leading zeroes and create identification numbers with same number of digits.

One final crucial step in data preprocessing consisted of the scaling of the data. Previously in this discussion we indicated that the relationship between variables is much more meaningful with scaled data. Learning machines also require scaled data to extract meaning and accurately learn and predict. Each variable, to include the response, was Mahalanobis scaled. After scaling the variables, the final step involved splitting the data into a training and testing set (4000 tuples were used for training, 1000 tuples for testing), and then the learning machines processed the information. Once the raw data is in MetaNeural format, $Analyze^{TM}$ operators perform and provide the scaling, splitting, and predictive modeling, and all necessary analytical results. This seven feature model achieved exceptional results, with an AUC of .907.

The technique tested next expands the details of the features in an attempt to improve the prediction. Each of the first six variables in the seven feature model is an average or percentage for the entire tuple of 100 commands considered. The seventh variable, x_u, represents the value which is a measure of the entire tuple. Therefore, each command can be represented by six different values, each value representing variables one through six. The result of this approach is a tuple of 601 features (the extra feature is Schonlau's x_u value). The results of this 601 feature model were surprisingly not better than the seven feature model. The 601 feature model was cumbersome because of its size, required much longer processing time, and did not predict as well. The best AUC achieved with the 601 feature model was .77.

The evaluation of several more preprocessing techniques further supported the idea of poor performance with expanded variables that could be represented as some type of average or summation for an entire tuple. Additional preprocessing techniques evaluated eliminated Schonlau's x_u value in an effort to eliminate the effect of this index-type feature. These additional preprocessing techniques are described in the below table.

It is intentional that the predictive modeling with the text mining variables

Table B.1: Preprocessing Techniques and Results

Preprocessing Technique	**Description of Variables**	σ	**AUC**
no_xu_value	Six variables without Schonlau's x_u value	1	.829
Enhanced Features (a)	Includes six variables plus an indicator for commands never seen from particular user	3	.87
Enhanced Features (b)	Same as (a) except user number (integer between 1 and 50) included as a variable	1	**.917**
Centralized Features I	Each tuple of test data contains 100 commands. This technique will examine only the center 32 commands of the tuple, and utilizing the six variables this will create 192 features for each tuple.	3	.714
Centralized Features II	Examine center 32 commands as discussed above, however produce aggregated six variables for every tuple.	1	.79
Rank Order	There are 635 distinct commands. Rank order from 1-635 based upon frequency. Now the tuple can be represented by a vector of 100 variables, with each feature being the number that represents the rank order of the command. The number 636 will represent any commands not seen in the training data.	1	.683
636feature	Utilize a vector of 636 variables for each tuple of data. Each position in the vector corresponds to the tuple's frequency of a distinct command, and position 636 will correspond to the frequency of foreign /never seen commands.	all values	.5

shown in Table B.1 are not compared against Schonlau's techniques. Schonlau builds the statistics for his models from the first 5,000 authentic commands from each user; this is an unsupervised approach. He tests his techniques with the subsequent 10,000 commands. The text mining techniques discussed in this paper focus solely on the subsequent 10,000 commands for each user and generally utilize a supervised approach. The text mining techniques are not comparable with Schonlau's techniques; they are fundamentally different.

The absence of Schonlau's x_u value created degraded performance as expected. Since the x_u value is an overall indicator of whether or not an intruder is present, it is a powerful predictor and hampers our clarity in determining the effectiveness of other variables. The best performance of the aforementioned predictive modeling occurred with Enhanced Features (b) with an AUC of .917 at $\sigma = 1$. The two

variables added to this preprocessing technique, commands never seen from the particular user (simply a 0 or 1 value) and the user number (integer between 1 and 50) provide an immense impact on the predictive power. It is likely that these two variables provide a personalization of each tuple, effectively providing a signature of each user that the learning machine identifies and uses to improve performance.

The other interesting thread of these results involves the continued poor performance of expanded preprocessing techniques that provide the learning machine with the raw data of each command as opposed to a summation or average. It is difficult to ascertain exactly why this phenomena occurs. During the discussion of the 601 feature model, we hypothesize that this degraded performance occurs because summations and averages are likely functions of sufficient statistics, and these sufficient statistics provide as much information as possible regarding the variable. The poor performance could also be an indicator that much noise and irrelevant data is digested by the learning machine, creating convoluted results.

B.2 Results of Combining Data Mining Techniques

During this research project, another student at Rensselaer Polytechnic Institute also designed variables for predictive modeling of Schonlau's data set. Yongqiang Zhang developed a method named recursive data mining (RDM) that is based on the simple but powerful model of cognition called a conceptor [135]. RDM recursively mines a string of symbols by finding frequent patterns, encoding them with unique symbols, and rewriting the string using this new coding. He used recursive data mining to characterize the structure and high-level symbols in user signatures and the monitored sessions.

In recursive data mining method, the input is encoded into symbols and then mined for dominant patterns. Dominant patterns are then assigned new codes that are then used to generate the data representation of a higher level. The input is thus recursively mined until no new dominant pattern exists. During the recursive mining, we generate several features that will be used to form the user's signature profile and will be compared to the features of monitored string. The result of this RDM technique was sixteen variables that could be merged with the text mining

variables.

In addition to the 9 text mining variables, we also utilized the first 16 variable from the Communal Intrusion Detection Technique of RDM. Figure B.1 illustrates the scaled values of these variables with respect to the output; this is the same analysis that we performed with the text mining variables in Figure 4.2. The output is again shown in the last column, and variables 1-25 are shown in th preceding columns. The first 9 variables are the text mining variables, and the remaining 16 variables are the RDM variables.

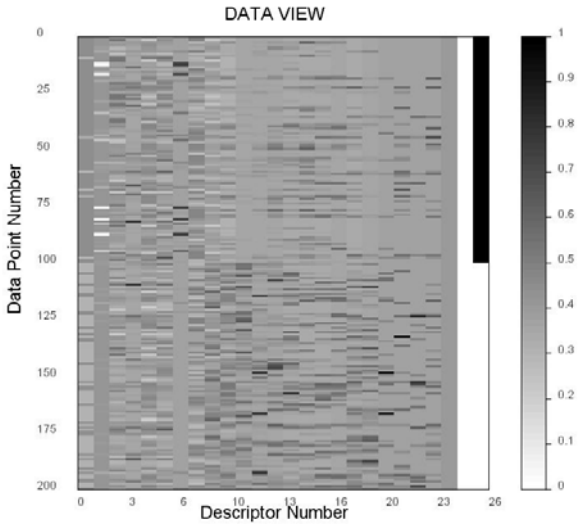

Figure B.1: Matrix plot of combined variables

The first variable in particular demonstrates predictive power, illustrating a clear direct correlation with the output. This is a text mining variable, "new", which is a 0 for any tuples that do not contain a new command never seen from a particular user and 1 if a new command is seen. A more descriptive method for representing this characteristic is to count the number of new commands. For example, in a tuple of 100 commands, it is probably not unusual to see one or two new commands, however some of the tuples contained 100 new commands! This is obviously a function of how the dataset was created (by inserting another user's data and not attempting to mimic and intruder). Using this variable alone as a prediction measure actually performs comparably with Schonlau's x_u value from the uniqueness algorithm. The x_u variable can be seen as the 9th column in Figure

B.1.

It is also of value to analyze the correlation that exists between variables, often identifying some of the "cousin" features that may be redundant. Within the text mining variables there is almost a perfect correlation that exists between the second and seventh variables; both of these variables involve the presence of unusual commands. Variable 2, "%Unix", captures the percentage of commands that are Unix. Variable 7, "foreign", captures the number of commands never seen before from the entire population of users. These are often non-Unix commands, resulting in the correlation.

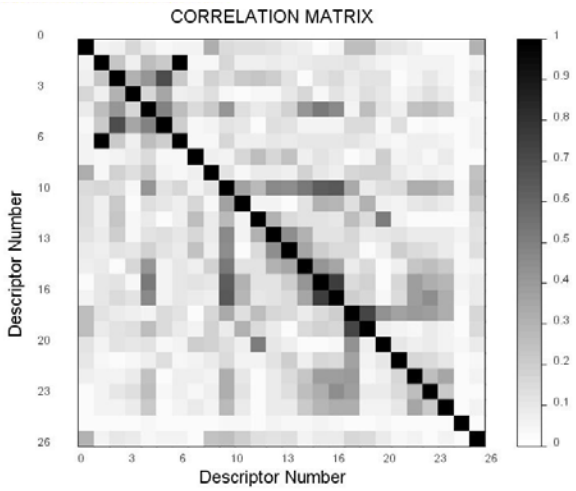

Figure B.2: Correlation of combined variables

The most valuable analysis that is evident from Figure B.2 is the minimal correlation between the text mining variables and the RDM variables. This minimal correlation indicates that combining these two preprocessing techniques should result in a synergistic effect, essentially outperforming the ROC convex hull (ROCCH) [56]. The ROCCH is the best operating point when considering the same False Positive tolerance for two classification systems. The goal is to combine systems somehow to exceed this ROCCH. Figure B.5 illustrates exactly this in a cartoon diagram.

After analyzing the fundamental statistics of the combined set of variables, we explored the predictive power of each variable set and the combined set. The learning machine utilized was Rosipal's Kernel Partial Least Squares (K-PLS) [124].

Utilizing a gaussian kernel element, choosing an appropriate value for σ was necessary. Through trial and error we found σ values of 1,2,or 3 typically worked best.

Similar to the text mining analysis discussed in section 4.5, the data was split into 4000 training tuples and 1000 testing tuples. The nine text mining variables (variables 1-9) achieve an area under the curve (AUC) of .944 utilizing the KPLS method. The sixteen RDM variables achieved and AUC of .915 (variables 10-25). Combining these two techniques achieves an AUC of .979. These results are shown in Figure B.3.

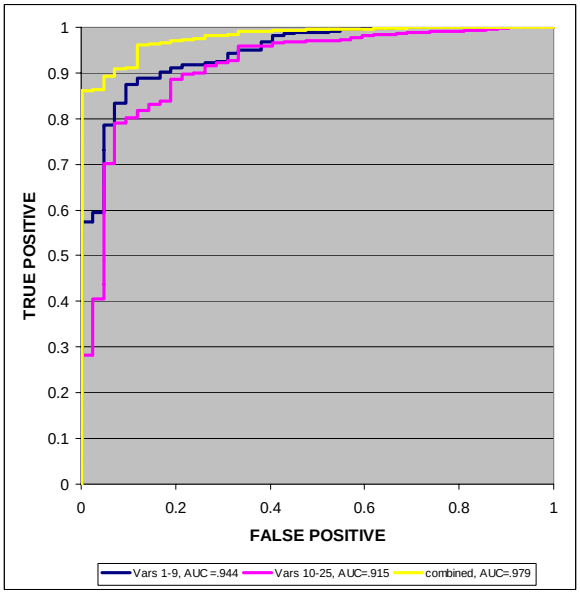

Figure B.3: ROC curve illustrating synergy from combination

The ROCCH of the subordinate curves would set almost directly on the curve of the text mining variables. There is obvious information that the learning machine gleans when combining these sets of variables that is not apparent from either of the sets when considered independently.

Combining sets of variates is a field that has been explored for many years, with much of the original work performed by Hotelling almost 75 years ago [78]. In the advent of learning machines, combinations of multiple classifiers is a term often mentioned. However, exploring the underlying statistics and illustrating the power of the combination through synergistic ROC curves is a novel concept. Future work in this field involves exploring the theoretical roots of what creates synergy when

combining two sets of variables.

B.3 Defining and Striving for Synergy

These next sections involve developing techniques in creating synergistic combinations of variables from Schonlau's dataset. It is the preprocessing of the data that sets the stage for a good or bad prediction model. During initial work with this data, our goal involved building combinations of classification systems that when combined exceeded the capabilities of either system alone - creating synergy. The measure of synergy comes from the analysis of each system's Receiver Operating Characteristic (ROC) curve and the ROC curve of the combined system. Fawcett and Provost explored combining multiple classification systems through the analysis of the ROC curves. Their research emphasized achieving optimality by operating on the convex hull produced by the ROC curves of the component systems, referred to as the ROCCH [56]. This research proposes that there is an optimum beyond the convex hull. By combining orthogonal sets of variables, synergy occurs.

A common yardstick to measure a binary classification system is the area under the ROC curve. The ROCCH of several classification systems is simply the curve representing the points of the greatest true positive ratings (from the component systems) for every possible false positive rating. If a combined classification system exceeds the ROCCH, it will inherently have a greater AUC and represent a synergistic classification system.

The below illustration is a cartoon sketch of a superior ROC curve that gains synergy from component systems.

One approach towards solving the optimal classification technique of the cartoon in Figure 1 is to determine simply where on the convex hull of the component systems there is an acceptable operating point. If cost prohibits multiple preprocessing techniques, this could perhaps be the best solution. However, when multiple preprocessing techniques are available, the prediction modeling discussed here apply.

The relevance of this work involves managing the selection of multiple preprocessing techniques or multiple sets of variables. This discussion is not about sensitivity analysis of individual variables; this discussion involves determining the

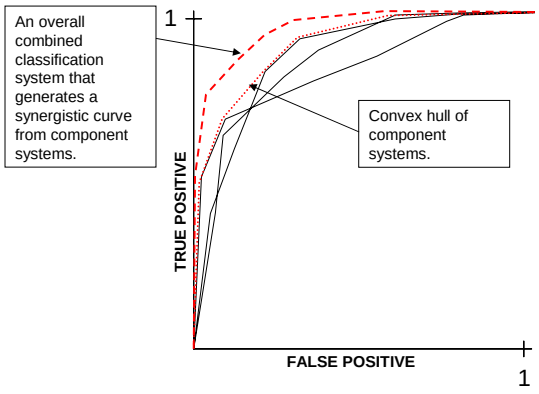

Figure B.4: Cartoon example of synergistic ROC curve

significance of a set of variables. There could be an instance when an organization must decide amongst several preprocessing systems or sets of variables. Cost, often measured in memory space or processing speed, may constrain the number of variable sets that an IDS can manage. These techniques provide quantitative and graphical techniques to illustrate the value of a set of variables. Decision makers who utilize these techniques in analyzing variable sets will provide themselves with improved opportunity for a successful IDS that operates as a superior binary classification system.

B.4 Orthogonal Preprocessing

The first step in any prediction modeling effort is determining how to preprocess the raw data. Since data preprocessing is often subjective and involves the data mining creativity of the researchers, it is difficult to quantify how to capture two or more orthogonal preprocessing techniques from the same set of data. From a semantic point of view, orthogonal preprocessing techniques occur by mining the same data set with different intentions. For example, there are two orthogonal preprocessing techniques explained in this report: text mining and recursive data mining. The underlying intentions behind the text mining involved describing a user's behavior by collecting statistics based upon both individual user and group command frequency, type, and popularity in many different manners and combinations. The underlying intentions of the recursive data mining involved detecting patterns of command usage for the purpose of identifying intruders and author identification

from the same data set [135]. These two preprocessing techniques significantly differ with regard to intended use, however they provide a measurement for common data. These two preprocessing techniques also demonstrate orthogonality, which will be shown later.

B.4.1 Analyzing The Text Mining Variables and RDM Variables

The text mining variables and RDM variables capture very different behavior involving the same data. The text mining variables focus on command types and the presence or absence of commands typically used by a particular user. The RDM variables represent patterns that exist within the data, and these patterns are relatively unique for each user, providing a measure of authenticity.

We observed good prediction modeling from both of these preprocessing techniques utilizing Rosipal's KPLS as a learning machine [124]. In an effort to maintain consistency and a basis for comparison, we conducted all of our prediction modeling with this learning machine. Since both techniques performed well for the same dataset, combining these variable sets seemed logical. Utilizing the area under the ROC curve (AUC) as a measure of performance, the combination of these variable sets achieved a noticeable improvement, shown in Figure B.5. The ROC curve of the combined variables exceeds the ROCCH of the component systems, illustrating synergy.

This synergy further encouraged the hypothesis that the two preprocessing techniques utilized possessed orthogonality. However, finding a measure of orthogonality would be necessary. There are two techniques utilized to determine this measure of orthogonality: the first technique involves analyzing the correlation that exists between the associated principal components of each set (ie, calculate the correlation of the text mining 1st principal component against the RDM 1st principal component, and the 2nd principal components, etc.) and the second technique exercised canonical correlation analysis.

B.4.1.1 Principal Components Analysis for Orthogonality

Principal components are a linear combination of the variables such that the 1st linear combination, or 1st principal component, possesses the maximum amount

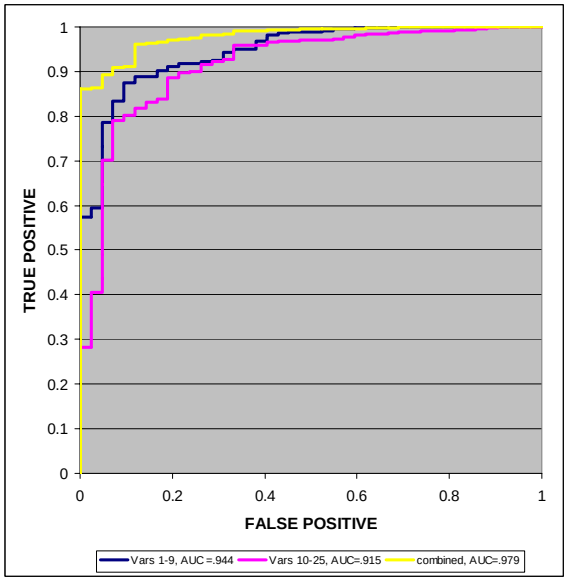

Figure B.5: Experimental results illustrating synergistic combination

of variance while remaining uncorrelated with the remaining principal components. Principal components analysis involves the decomposition of the correlation matrix; see [85] for a more detailed explanation.

Let $\rho_{i,j}$ represent the correlation between the ith principal component of the text mining variables and the jth principal component of the recursive data mining variables. If the respective principal components of these variable sets presented a low correlation ($\mid \rho_{i,j} \mid < .5$), especially for principal components that represent greater that 10% of the variance, then we consider these variable sets orthogonal and expect synergy from the combination of the variables. The color correlation plot of the first three text mining principal components and the first three RDM principal components is shown in Figure B.6.

In an effort to create a non-synergistic combination, we compared the principal components of a subset of variables from the text mining variables against the principal components from the entire set of text mining variables. As one could expect based upon this experiment, the principal components demonstrated a high correlation, where $\mid \rho_{i,j} \mid > .8$ in some cases. For this example, the ROC curve of the combination sat almost directly on top of the ROCCH of the two component systems.

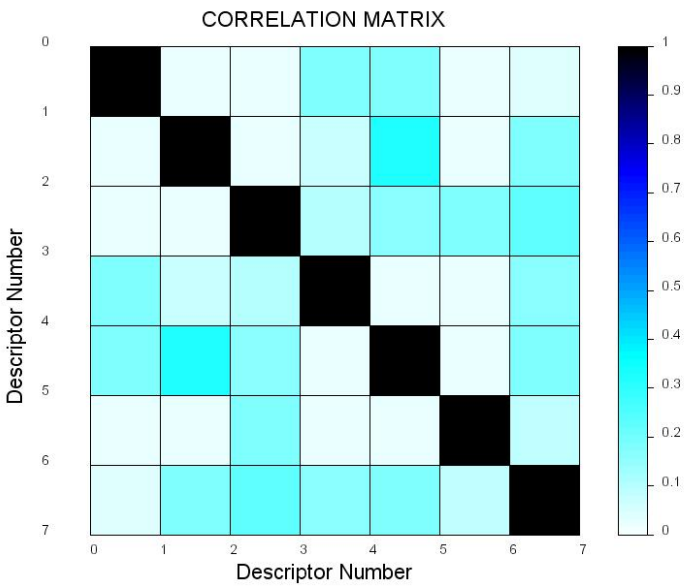

Figure B.6: Color correlation of first 3 principal components between text mining variables and RDM variables

Although lower correlations between principal components of different variable sets provided an indication for orthogonality, it is possible that high correlations could result from two data sets that create synergy when combined. If there are rogue variables that create a high degree of variance within a data set but provide minimal predictive power, these variables will carry a strong weight in the principal components. If both sets of variables have such a rogue variable, and these rogue variables correlate for some reason, high correlation could result between principal components of orthogonal variable sets. For this reason, it is important to always Mahalanobis scale (normalize data, subtracting the sample mean and dividing by the sample standard deviation) data first and compare the weights (coefficients) that exist in the linear combinations that represent the scores for the principal components. If these weights strongly favor these rogue type variables, further analysis of the orthogonality is necessary. Another technique to examine orthogonality is canonical correlation analysis.

B.4.1.2 Canonical Correlation Analysis for Orthogonality

Canonical correlation analysis is multivariate statistical technique designed for comparing two sets of variables. The background for this technique is discussed in

section 2.2.1.2.

Similar to principal components analysis, the first several canonical correlations provide the most significant information. We observed the 1st canonical correlation between the text mining variables and the RDM variables to be .699, which would be high when considering two variables. However, we observed synergy through this combination. Through our experimental results, we observed synergy for when the absolute value of the first canonical correlations was generally lower than .8.

B.5 Experimental Results for Orthogonal Analysis

There were three cases examined in the experiment performed with the principal components. In regard to the numbering of the variables, recall that variables 1-9 are the text mining variables and variables 10-25 are the RDM variables. We created three different cases: Case 1 compared variables 1,7,8,9 against variables 1-9. This is an instance where one of the variable sets is a subset of the other variable set. High correlation between the principal components of each variable set is expected, and this is what occurred.

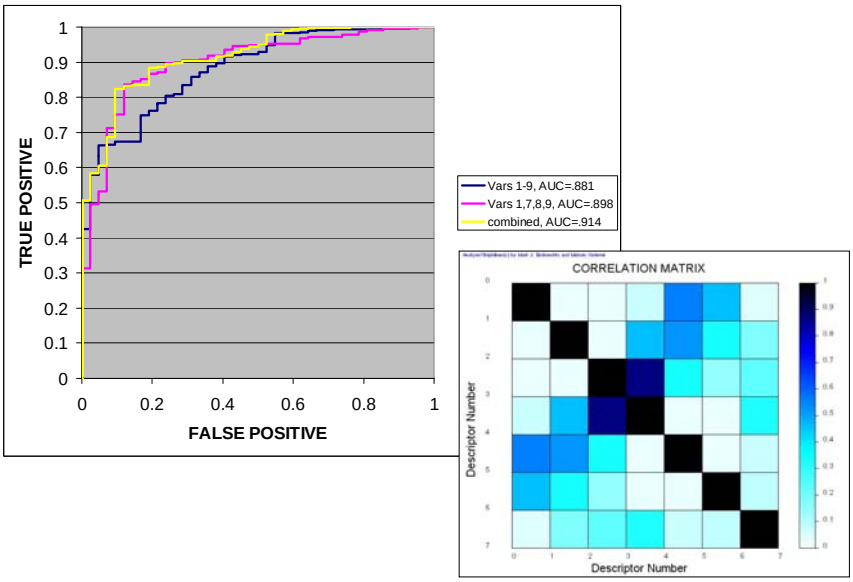

Figure B.7: ROC curves and correlation matrix from Case 1

The ROC curves in Figure B.7 illustrate the performance of Rosipal's KPLS

when the features are the principal components of the variables indicated in the legend. For example, for the line representing variables 1,7,8,9, this ROC curve is a result of using the first three principal components of these four variables as features. For the ROC curve representing the combined model, this simply involved concatenating the three principle components from each variable set to create a prediction model with six features. The purpose of analyzing the combined model is to determine if the combination creates synergy. The idea with Case 1 is to show that if the principle components between two sets of variables show a high correlation, the combination of these variable sets will not produce a synergistic ROC curve. As a matter of fact, this combination produced an ROC curve that sat almost directly on the ROCCH.

Case 2 compared variables 1,7,8,9 against variables 2-6. This case involved all text mining variables, but the variable sets were mutually exclusive (ie, the same variable was not in both sets). The results of this case are shown in Figure B.8.

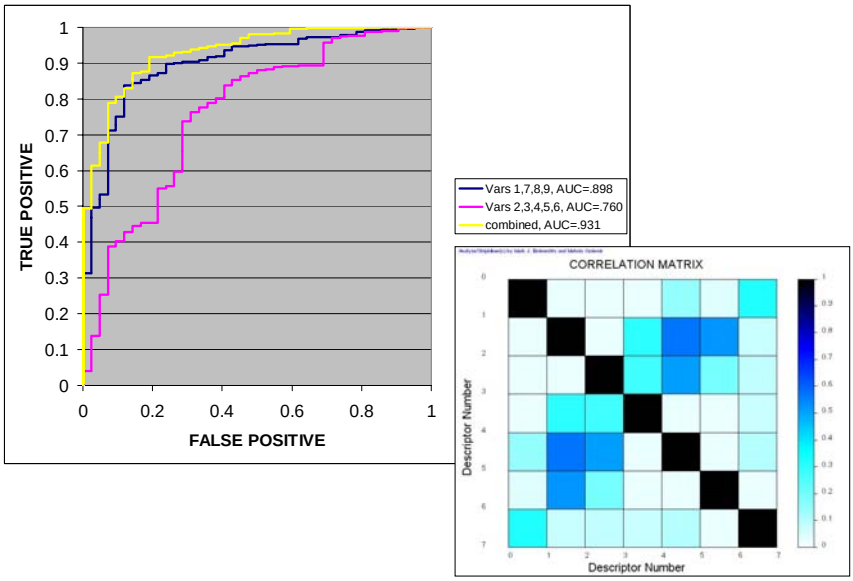

Figure B.8: ROC curves and correlation matrix from Case 2

Case 2 is beginning to show synergy. Notice that the correlation of the variables is lower, and the ROC curve of the combined principle components lie noticeably outside of the ROCCH.

Case 3 compared the text mining variables versus the RDM variables, or vari-

ables 1-9 against variables 10-25. Figure B.9 illustrates the results from this case.

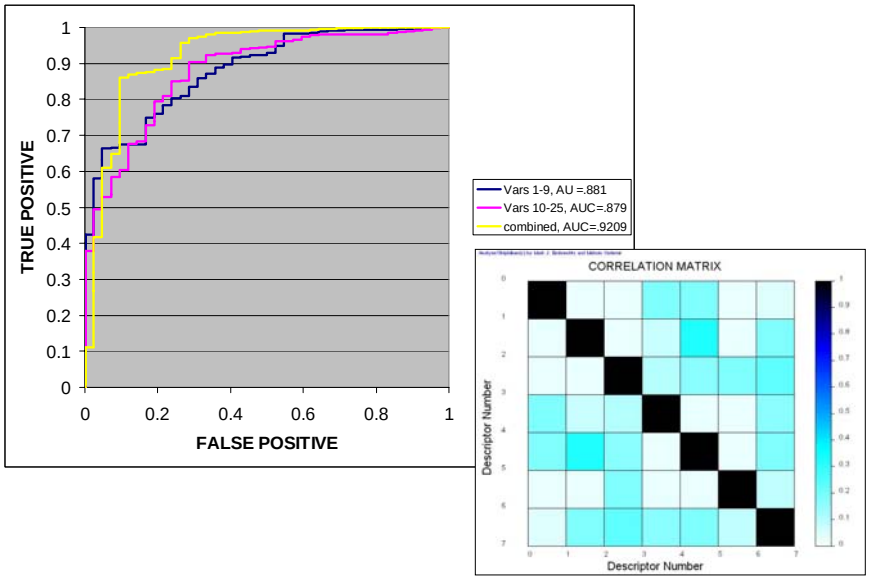

Figure B.9: ROC curves and correlation matrix from Case 3

This case illustrates a noticeable increase in the ROC curve for the combined principal components. The correlations between the principal components from each set of variables is low; the highest correlation is .34.

The canonical correlation analysis also provided encouraging results. Since the 1st canonical correlation always has the largest correlation value, a low value for this 1st canonical correlation represents orthogonal sets of variables which should produce a synergistic combination. Analyzing the canonical correlation for each of the same three cases considered for principal components analysis provided the following results:

Table B.2: Canonical Correlation Analysis Results

	1st Canonical Correlation	**2nd Canonical Correlation**	**3rd Canonical Correlation**
Case 1	.6999	.5597	.4536
Case 2	1	1	1
Case 3	.3941	.2971	.1537

Both case 1 and case 3 demonstrated synergy when combined. Case 2 did not demonstrate synergy, as expected, because one set of variables was simply a subset of the other set of variables. Given two sets of variables that predict reasonably well (AUC ¿ .7), combining these sets of variables prior to prediction modeling will create synergistic improvement given a low canonical correlation between the sets of variables.

The analysis of synergy between sets of variables provides a method to evaluate the contribution that a set of variables provides to a prediction machine. This research directly relates to sensitivity analysis and provides a platform to summarize results when multiple sets of variables exist.

Much of today's research in intrusion detection systems involves examining multiple intrusion detection systems, hybrid systems, and ensemble methods. When collecting and considering a combination of agents that will provide intrusion detection or prevention, analyzing the synergy of the variable sets provides a platform for deciding which agents provide redundant performance. In addition to providing a higher level of analysis for combinations of multiple classifiers involving intrusion detection, the techniques previously discussed could extend to almost any classification problem involving multiple sets of variables. The ROC curve analysis is particularly applicable to binary classifications problems. Security problems in general can always be modeled as a binary classification problem, and the improvement of our society's security posture is an endeavor worthy of continued pursuit.

APPENDIX C
IMPLEMENTATION OF GENETIC ALGORITHM IN PERL

```
########################################
# Genetic Algorithm written by
# Paul F. Evangelista
# DEC 04
#
# This program utilizes a genetic algorithm
# to find uncorrelated subspaces by measuring
# their principal component correlation (across
# subspaces)
########################################


sub shuffle  #Randomly shuffle any given array
{
my $array = shift;
my $i;
for ($i=@$array;--$i;)
{
my $j = int rand ($i+1);
next if $i == $j;
@$array[$i,$j] = @$array[$j,$i];
}
}

sub fitness  {  #Calculates fitness of each member utilizing Analyze
my @rand_chrom = @_;
our $gen_size;
my ($i,$k,@array,@subset_1,@subset_2,@subset_3,$call,@fitness);
for ($k=0;$k<$gen_size;$k++)  {
@array = split /\t/,$rand_chrom[$k];
```

```
#print "chrom$k:  @array\n";

for ($i=0;$i<9;$i++)  {
#creates subsets and eventually writes them to file
$subset_1[$i] = $array[$i];
$subset_2[$i] = $array[$i+9];
if ($i<8)  {
$subset_3[$i]=$array[$i+18];
}
}

@subset_1 = sort{$a<=>$b} @subset_1;
@subset_2 = sort{$a<=>$b} @subset_2;
@subset_3 = sort{$a<=>$b} @subset_3;

open(set_1,">set_1.txt");
open(set_2,">set_2.txt");
open(set_3,">set_3.txt");

for ($i=0;$i<9;$i++)  {
print set_1 "$subset_1[$i]\n";
print set_2 "$subset_2[$i]\n";
$subset_2[$i] = $array[$i+9];
if ($i<8)  {
print set_3 "$subset_3[$i]\n";
}
}
#$call = "sh W_SVM.bashrc > details.txt";
$call = "sh W_distance.bashrc > details.txt";
system $call;
open(fit,"W.txt");
$fitness[$k] = <fit>;
$fitness[$k] = -1*$fitness[$k]; ################## FOR MAX W
}
return @fitness;
}

sub sel_breed {  #receives array of fitness values
```

```
my @input = @_;
my $length = @input;
my ($i,@prob,$rn,$j,@sel_gen,$comp,@sort_input,@new_input);
my $cum_prob =0;
my $sum = 0;
my @new_input = sort{$a<=>$b} @input;
$new_input[$length] = $new_input[$length-1] + 1;
for ($i=0;$i<$length;$i++)  {
#creates sort input which is an index array to original input in sorted order
$comp = 0;
for ($j=0;$j<$length;$j++)  {
if ($input[$j] == $new_input[$i]) {
$sort_input[$i] = $j;
}
}
}

  my $maxer = 0;
for ($i=0;$i<$length;$i++)  { #find max fitness of current generation
if ($new_input[$i] > $maxer)  {
$maxer = $new_input[$i];
}
}
$maxer = $maxer+.1;
for ($i=0;$i<$length;$i++)  { #Create probabilities for selection
$sum = $sum + ($maxer - $new_input[$i]);
}

for ($i=0;$i<$length;$i++)  { #Create probabilities for selection
$prob[$i] = ($maxer-$new_input[$i])/$sum;
#adjusts probability for minimization problem
$cum_prob = $cum_prob + $prob[$i];
$prob[$i] = $cum_prob;
print "$sort_input[$i] $new_input[$i] $prob[$i]\n";
}

for ($i=0;$i<$length-1;$i++)  {  #Select next generation
$rn = rand;
```

```
$prob[$length] = 1.1;
for ($j=0;$j<$length;$j++)  {
if ($j==0)  {
if ($rn > 0 && $rn <= $prob[$j])  {
$sel_gen[$i] = $sort_input[$j];
last;
}
}

if ($j>0)  {
if ($rn > $prob[$j-1] && $rn <= $prob[$j])  {
$sel_gen[$i] = $sort_input[$j];
last;
}
}
}
#print "rand#:  $rn index: $sel_gen[$i]\n";
}
$sel_gen[$length-1] = $sort_input[0]; #elitist selection
print "Elite member:  $sort_input[0] fitness:  $input[$sort_input[0]]\n";
return @sel_gen;
}

our @ordered_array;

for ($i=0;$i<26;$i++)  {
$ordered_array[$i] = $i+1;
}

for ($i=0;$i<50;$i++)  { #create a random number of initial members
@array = @ordered_array;
shuffle(\@array);
$rand_chrom[$i] = join "\t",@array;
}

############################################################

our $no_gens = 30; #Experimental Control parameters
```

```
our $gen_size = 30;
$worst_fitness = 0;

############################################################

@current_chrom = @rand_chrom;

for ($zz=0;$zz<$no_gens;$zz++)  {

@pop_fitness = fitness(@current_chrom);
$best_fitness = 1;
#best fitness will range from 0 to 1, where 0 is best
print "fitness of generation:\n";
for ($i=0;$i<$gen_size;$i++)  {
print " $pop_fitness[$i]\n";
#print "$current_chrom[$i]\n";
if ($pop_fitness[$i] < $best_fitness)  {
$best_fitness = $pop_fitness[$i];
$fittest = $current_chrom[$i];
}
if ($pop_fitness[$i] > $worst_fitness)  {
$worst_fitness = $pop_fitness[$i];
$worst = $current_chrom[$i];
}
}
print "CURRENT BEST FITNESS:  $best_fitness\nFITTEST CHROMOSOME:  $fittest\n";
print "\n";

@breeders = sel_breed(@pop_fitness);

@array = @ordered_array;
shuffle(\@array); #insert two immigrants for worst performers
$curr_chrom[$breeders[$gen_size-3]] = join "\t",@array;
shuffle(\@array);
$curr_chrom[$breeders[$gen_size-2]] = join "\t",@array;

print "the breeders (elitist is last one):\n @breeders\n";
$elitist = $breeders[$gen_size-1];
```

```
$elite_chrom = $current_chrom[$elitist];
print "elite at 158:  $elite_chrom\n";

pop(@breeders);
shuffle(\@breeders);

$k=0;
$l=0;
for ($i=0;$i<($gen_size-1);$i++)  {
$rn = rand;
if ($rn > .4)  {
$crossover[$k] = $i;
$k++;
}
if ($rn <= .4)  {
$new_chrom[$l] = $current_chrom[$breeder[$i]];
$l++;
}
}



shuffle(\@crossover);

$x_over_pu = $l;
$no_xover = $k;

for ($i=0;$i<$no_xover;$i=$i+2)  {
#print "creating offspring from parents $crossover[$i] and $crossover[$i+1]\n";
@parent_1 = split /\t/,$current_chrom[$crossover[$i]];
@parent_2 = split /\t/,$current_chrom[$crossover[$i+1]];
$length = @array;
$c_over_point = int(@array/2);
for ($j=0;$j<$c_over_point;$j++) {
$child_1[$j] = $parent_1[$j];
$child_2[$j] = $parent_2[$j];
}
for ($j=$c_over_point;$j<$length;$j++) {
```

```
$child_1[$j] = $parent_2[$j];
$child_2[$j] = $parent_1[$j];
}
$m=0;
$n=0;
for ($j=0;$j<$length;$j++)  {
$dup_1[$j] = 0;
$dup_1[$j] = 0;
$used_1 = 0;
$used_2 = 0;
for ($l=0;$l<$length;$l++)  {
#checks to see what integers went unused and assigns them to an "avail" array
if ($child_1[$l] == $j+1)  {
$used_1 = 1;
}
if ($child_2[$l] == $j+1)  {
$used_2 = 1;
}
}
if ($used_1 == 0)  {
$avail_1[$m] = $j+1;
$m++;
}
if ($used_2 == 0)  {
$avail_2[$n] = $j+1;
$n++;
}
}
$m=0;
$n=0;
for ($j=0;$j<$c_over_point;$j++) {
for ($l=$c_over_point;$l<$length;$l++)  {
if ($child_1[$j] == $child_1[$l])  {
$child_1[$j] = $avail_1[$m];
$m++;
}
if ($child_2[$j] == $child_2[$l])  {
$child_2[$j] = $avail_2[$n];
```

```
$n++;
}
}
}
@check1 = sort{$a<=>$b} @child_1;
@check2 = sort{$a<=>$b} @child_2;
#print "child_1 sorted after check:\n  @check1\n";
#print "child_2 sorted after check:\n  @check2\n";

$new_chrom[$x_over_pu] = join "\t", @child_1;
$x_over_pu++;
$new_chrom[$x_over_pu] = join "\t", @child_2;
$x_over_pu++;
}



print "\n\n";
$size = @new_chrom;
print "size should be gen_size:  $size\n";

for ($l=0;$l<$gen_size-1;$l++)  { #MUTATION
for ($k=0;$k<$length;$k++)  {
$rn = rand;
if ($rn < .01)  {
#print "chrom before mutation:\n  $new_chrom[$l]\n";
@array = split /\t/, $new_chrom[$l];
$mute = $array[$k];
$mute_a = $array[$length-$k];
$array[$k] = $mute_a;
$array[$length-$k] = $mute;
$new_chrom[$l] = join "\t", @array;
#print "chrom after mutation:\n  $new_chrom[$l]\n";
}
}
#print "$new_chrom[$l]\n\n";
$current_chrom[$l] = $new_chrom[$l];
}
$current_chrom[$gen_size-1] = $elite_chrom;
```

```
print "elitist at end of program:  \n $current_chrom[$gen_size-1]\n";

}
print "Worst fitness is:  $worst_fitness\n\n$worst\n\n";

print "Best fitness is : $best_fitness\n\nFittest chromosome:\n$fittest\n";

open (worst_set1,">worst_set1.txt");
open (worst_set2,">worst_set2.txt");
open (worst_set3,">worst_set3.txt");
open (best_set1,">best_set1.txt");
open (best_set2,">best_set2.txt");
open (best_set3,">best_set3.txt");
@worst_sets = split /\t/,$worst;
@best_sets = split /\t/,$fittest;
@length = @worst_sets;


for ($i=0;$i<$length;$i++)  {
if ($i<9)  {
print worst_set1 "$worst_sets[$i]\n";
print best_set1 "$best_sets[$i]\n";

}
if ($i<18 && $i>=9)  {
print worst_set2 "$worst_sets[$i]\n";
print best_set2 "$best_sets[$i]\n";
}
if ($i>=18)  {
print worst_set3 "$worst_sets[$i]\n";
print best_set3 "$best_sets[$i]\n";
}
}



open (GAout, ">GAout.txt");
print GAout "Worst fitness is:  $worst_fitness\n\n$worst\n\n";
```

```
print GAout "Best fitness is : $best_fitness\n\nFittest chromosome:\n$fittest\n";

##########  END OF MAIN LOOP ####################################


print "Worst fitness is:  $worst_fitness\n\n$worst\n\n";
print "Best fitness is : $best_fitness\n\nFittest chromosome:\n$fittest\n";

@array = split /\t/,$fittest;
for ($i=0;$i<9;$i++)  {  #creates subsets and eventually writes them to file
$subset_1[$i] = $array[$i];
$subset_2[$i] = $array[$i+9];
if ($i<8)  {
$subset_3[$i]=$array[$i+18];
}
}

@subset_1 = sort{$a<=>$b} @subset_1;
@subset_2 = sort{$a<=>$b} @subset_2;
@subset_3 = sort{$a<=>$b} @subset_3;

open(set_1,">set_1.txt");
open(set_2,">set_2.txt");
open(set_3,">set_3.txt");

for ($i=0;$i<9;$i++)  {
print set_1 "$subset_1[$i]\n";
print set_2 "$subset_2[$i]\n";
$subset_2[$i] = $array[$i+9];
if ($i<8)  {
print set_3 "$subset_3[$i]\n";
}
}
```